TUGAS AKHIR – TI 141501

PENERAPAN ALGORITMA PARTICLES SWARM OPTIMIZATION DALAM PENYELESAIAN GATE ASSIGNMENT PROBLEM *(STUDI KASUS : BANDARA SOEKARNO-HATTA)*

HENDRIK FEBRIYANTO
NRP 02411240000061

Dosen Pembimbing
Prof. Ir. Budi Santosa M.S. Ph.D

JURUSAN TEKNIK INDUSTRI
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2018

LEMBAR PENGESAHAN

PENERAPAN ALGORITMA PARTICLES SWARM OPTIMIZATION DALAM PENYELESAIAN GATE ASSIGNMENT PROBLEM *(STUDI KASUS : BANDARA SOEKARNO-HATTA)*

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat Memperoleh Gelar Sarjana Teknik
pada
Program Studi S-1 Jurusan Teknik Industri
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya

Oleh :
HENDRIK FEBRIYANTO
NRP 02411240000061

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Prof. Ir. Budi Santosa, M.S., Ph.D.
NIP. 196905121994021001

SURABAYA, JANUARI 2018

PENERAPAN ALGORITMA PARTICLES SWARM OPTIMIZATION DALAM PENYELESAIAN GATE ASSIGNMENT PROBLEM *(STUDI KASUS : BANDARA SOEKARNO-HATTA)*

Nama : Hendrik Febriyanto

NRP : 02411240000061

Pembimbing : Prof. Ir. Budi Santosa, M.S., Ph.D

ABSTRAK

Gate Assignment bertujuan untuk menetapkan *flight* yang ada pada *gate* yang tersedia sehingga mampu meminimalkan ketidaknyamanan penumpang dan biaya operasional bandara serta maskapai penerbangan. *Gate Assignment Problem* (GAP) dapat didefinisikan sebagai penetapan penugasan dari *flight* yang ada menuju *gate* yang tersedia dengan mengoptimalkan berbagai ukuran performansi yang menjadi batasan. Sebagian besar, fungsi tujuannya adalah meminimalisasi total jarak penumpang berjalan kaki atau yang disebut *total passenger walking distance*. Hal yang perlu diperhatikan dalam permasalahan ini merupakan jumlah penumpang yang melakukan penerbangan transit, jumlah penumpang yang tiba dan jumlah penumpang yang berangkat, jarak antar *gate* dan jarak antara *gate* dengan *arrival* dan *departure hall*. Dalam penelitian ini untuk mencari total *passenger walking distance* digunakan algoritma *Particle Swarm Optimization* (PSO). PSO didasarkan pada perilaku sebuah kawanan burung atau ikan, dimana setiap individu atau partikel berperilaku dengan cara menggunakan kecerdasannya sendiri dan juga dipengaruhi perilaku kelompok kolektifnya. Alasan lain mengapa PSO layak diterapkan pada berbagai permasalahan adalah bahwa terdapat beberapa parameter yang dapat disesuaikan. Satu versi, dengan sedikit variasi, bekerja dengan baik dalam berbagai macam aplikasi. Hasil akhir dari penelitian ini adalah didapatkan total *passenger walking distance* sebesar 21339 kilometer dengan 1000 iterasi yang berjalan selama sekitar 6 detik. Dengan waktu komputasi yang relatif singkat didapatkan solusi yang *feasible* menjadi tolak ukur bahwa PSO dapat digunakan dalam menyelesaikan GAP.

Kata kunci : *Gate Assignment, Particles Swarm Optimization, total passenger walking distance*

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa sehingga penulis dapat menyelesaikan pengerjaan Tugas Akhir dengan judul **"Penerapan Algoritma Particles Swarm Optimization Dalam Penyelesaian Gate Assignment Problem *(Studi Kasus : Bandara Soekarno-Hatta)"***.

Penulisan Tugas Akhir ini dilakukan untuk memenuhi persyaratan menyelesaikan studi Strata-1 (S1) dan memperoleh gelar Sarjana Teknik Industri di Jurusan Teknik Industri, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya.

Selama penulisan Tugas Akhir ini, penulis mendapatkan berbagai bantuan dan dukungan dari berbagai pihak. Pada kesempatan ini, penulis mengucapkan terima kasih kepada :

1. Bapak Prof. Ir. Budi Santosa, M.S., Ph.D selaku Dosen Pembimbing yang telah sabar menuntun serta memberi saran dan kritik selama proses pengerjaan Tugas Akhir.
2. Saudara Siti Dwi Rahmawati, S.T. selaku penyedia data.
3. Ibu Effi Latifianti, M.Sc, Bapak Dody Hartanto, S.T., M.T., dan Bapak Yudha Andrian Saputra, S.T., MBA. selaku dosen penguji seminar dan sidang Tugas Akhir.
4. Ketua Departemen, Sekretaris Departemen, Kepala Program Studi, dan Sekretaris Program Studi Departemen Teknik Industri ITS.
5. Seluruh stakeholder Departemen Teknik Industri ITS. Para Dosen yang telah mengampu penulis sehingga mendapatkan ilmu yang nantinya pasti berguna dalam kehidupan.
6. Keluarga penulis, terutama kedua orang tua penulis yaitu Bapak Anwar dan Ibu Wantini, serta kakak kandung penulis Hendra Yulian. Keluarga penulis yang selalu memberikan dukungan, doa dan semangat sehingga penulis mampu menuntaskan penulisan Tugas Akhir ini. Keluarga besar UKM Bridge dan Lembaga Minat Bakat ITS yang selalu mendukung penulis untuk selalu berprestasi selama menjadi mahasiswa di ITS. Keluarga besar KAVALERI yang senantiasa memberikan dukungan dan semangat kebersamaan.

Serta berbagai pihak yang tidak dapat penulis sebut satu persatu, terima kasih atas semua dukungan, nasihat, semangat yang diberikan kepada penulis. Semoga Tuhan selalu memberkati kita semuanya.

Selama pengerjaan Tugas Akhir ini, penulis menyadari bahwa penulisan Tugas Akhir ini masih jauh dari kata sempurna. Saran dan kritik sangatlah membantu untuk proses ke depannya. Penulis berharap Tugas Akhir ini mampu memberikan manfaat bagi para pembacanya. Demikian yang dapat penulis sampaikan, akhir kata penulis menyampaikan terima kasih.

Surabaya, Januari 2018

Hendrik Febriyanto

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

BAB 1
PENDAHULUAN

Pada bab pendahuluan dijelaskan mengenai hal-hal yang mendasari dilakukannya penelitian serta identifikasi masalah penelitian yang meliputi latar belakang, perumusan masalah, tujuan penelitian, manfaat penelitian, serta batasan dan asumsi yang digunakan dalam penelitian.

1.1 Latar Belakang

Bisnis penerbangan saat ini sudah berkembang pesat. Bisnis penerbangan memiliki tiga aktor utama yang berperan yakni Industri Manufaktur Pesawat Terbang, Maskapai Penerbangan dan Bandar Udara (Bandara) (Wittmer et al., 2011). Ketiga aktor utama ini berperan saling melengkapi satu sama lain. Dimana Industri Manufaktur sebagai penyedia pesawat, Maskapai berperan dalam hal sistematisasi bisnis penerbangan, sedangkan Bandara merupakan sarana yang tak dapat dilepaskan terkait bisnis penerbangan ini. Adanya pesawat tanpa adanya sistem bisnis yang baik akan membuat pesawat yang ada tidak dapat menghasilkan *added value* berlebih. Begitu pula meskipun adanya sistem yang baik namun tidak adanya pesawat maka sistem yang dibuat juga percuma adanya karena tidak ada prasarana yang dapat dipergunakan. Ada pesawat dan sistem yang baik namun tidak ditopang dengan bandara di lokasi bisnis maka semuanya akan sia-sia.

Dalam tulisan ini fokus yang diambil terkait dengan Bandara. Beberapa masalah serius terkait dengan operasi bandara antara lain penggunaan kapasitas landasan pacu, penundaan penerbangan, pencemaran lingkungan akibat kebisingan dan emisi gas, dan juga keamanan (Dell'Orco et al., 2017). Dari berbagai masalah tersebut, terdapat masalah yang menarik dimana terdapat cukup banyak jadwal penerbangan yang ada dalam sebuah bandara sehingga pengaturan yang tepat terkait dengan pengaturan penumpang dan bagasi menjadi salah satu hal mendasar yang memiliki pengaruh tinggi. Meskipun biaya aktivitas ini umumnya hanyalah porsi kecil dari keseluruhan biaya operasi maskapai penerbangan, namun memiliki dampak yang sangat besar dalam menjaga efisiensi jadwal penerbangan dan kepuasan penumpang (Bazargan, 2010). Dengan meningkatnya lalu lintas udara sipil dan pertumbuhan bandara dalam beberapa dekade terakhir menyebabkan kompleksitas penugasan ini meningkat secara signifikan. *Gate assignment* telah menjadi salah satu aktivitas utama dalam operasi bandara.

Manajemen pengelolaan bandara memiliki masalah yang cukup kompleks. Salah satu tugas penting di bandara adalah menetapkan *flight* untuk ditugaskan pada *gate* yang tersedia. Pengelolaan penunjukan *gate* yang lebih efisien akan menghasilkan *delay flight* yang lebih rendah, tingkat penggunaan fasilitas *ground* yang lebih efektif dan tentu saja layanan penumpang yang lebih baik. Untuk itu jelas, dengan pengaturan ini akan menyebabkan biaya operasi lebih rendah. Selain itu,

didapatkan pula nilai lebih yakni kepuasan penumpang akan meningkat. Terkadang, jumlah *flight* tidak selaras dengan jumlah *gate* yang tersedia, sehingga dalam hal ini, manajemen harus mampu meminimalkan jumlah *flight* yang tidak beraturan.

Gate Assignment Problem (GAP) dapat didefinisikan sebagai penetapan penugasan dari *flight* yang ada untuk ditugaskan menuju *gate* yang tersedia dengan mengoptimalkan berbagai ukuran performansi yang menjadi batasan. Sebagian besar, GAP meminimalisasi total jarak penumpang berjalan kaki. Jarak yang ditempuh penumpang dapat dibedakan yakni jarak dari loket masuk (gerbang *check-in*) menuju *gate*, jarak dari *gate* menuju area bagasi (*baggage claim*) dan/atau loket keluar dan jarak dari satu *gate* menuju *gate* lainnya bagi penumpang transit (Aktel et all, 2016).

Pada penelitian-penelitian yang telah dilakukan sebelumnya disampaikan bahwa permasalahan GAP ini cukup penting untuk diatasi mengingat *gate* merupakan salah satu kunci utama dalam operasi bandara. Pengaturan *gate* yang baik dapat merealisasikan proses parkir di *gate* yang dituju atau yang disebut *docking* dengan lebih cepat dan tepat. Selain itu pula *gate assignment* yang baik juga akan mampu memastikan koneksi yang efektif antara *flight* dan bandara sehingga mampu meningkatkan kapasitas dan efisiensi layanan di bandara. Kasus GAP kerap kali diselesaikan dengan berbagai metode optimasi baik dengan *linear programing* maupun dengan heuristik. Untuk metode heuristik sebagian peneliti juga sudah cukup banyak mencoba dengan menggunakan metode metaheuristik dengan menggunakan berbagai algoritma, seperti *Genetic Algorithm* (GA), *Tabu Search*, *Fuzzy Bee Colony*, *Simulated Annealing*, *Greedy-A* dan mungkin masih ada yang lainnya. Metode metaheuristik cukup sesuai bila diterapkan dalam menyelesaikan GAP ini karena tergolong dalam permasalahan yang memiliki cukup *constraint* dan variabel serta data yang banyak dalam setiap bandara. Dengan banyaknya data maka variabel akan cukup banyak sehingga penerapan metode metaheuristik cukuplah sesuai mengingat metaheuristik tidak memakan waktu cukup banyak dalam komputasinya.

Algoritma yang digunakan dalam menyelesaikan GAP kali ini adalah *Particle Swarm Optimization* (PSO). PSO sendiri didasarkan pada perilaku sebuah kawanan burung atau ikan. Setiap individu atau partikel berperilaku dengan cara menggunakan kecerdasannya sendiri dan juga dipengaruhi perilaku kelompok kolektifnya. Dengan demikian, jika satu partikel atau seekor burung menemukan jalan yang tepat atau pendek menuju sumber makan, maka sisa kelompok yang lain juga akan dapat segera mengikuti jalan tersebut meskipun lokasi mereka jauh di kelompok tersebut. PSO telah diterapkan untuk menyelesaikan berbagai persoalan optimasi seperti VRP, penjadwalan proyek kerja dan beberapa masalah lainnya. Pada algoritma PSO ini, pencarian solusi dilakukan oleh suatu populasi yang terdiri dari beberapa partikel. Populasi dibangkitkan secara random dengan batasan nilai terkecil dan terbesar. Setiap partikel merepresentasikan posisi atau solusi dari permasalahan yang dihadapi. Setiap partikel merepresentasikan posisi atau solusi yang optimal

dengan melintasi ruang pencarian. Hal ini dilakukan dengan cara setiap partikel melakukan penyesuaian terhadap posisi terbaik dari partikel tersebut (*local best*) dan penyesuaian terhadap posisi partikel terbaik dari seluruh kawanan (*global best*) selama melintasi ruang pencarian. Jadi, penyebaran pengalaman atau informasi terjadi di dalam partikel itu sendiri dan antara suatu partikel dengan partikel terbaik dari seluruh kawanan selama proses pencarian solusi. Setelah itu, dilakukan proses pencarian untuk mencari posisi terbaik setiap partikel dalam sejumlah iterasi tertentu sampai didapatkan posisi yang relatif *steady* atau mencapai batas iterasi yang ditetapkan. Pada setiap iterasi, setiap solusi yang direpresentasikan oleh posisi partikel, dievaluasi performansinya dengan cara memasukkan solusi tersebut ke dalam *fitness function* (Santosa & Willy, 2011).

1.2 Rumusan Masalah

Permasalahan yang akan dibahas pada penelitian tugas akhir ini adalah bagaimana menentukan penugasan *flight* dalam *gate* yang tersedia menggunakan algoritma *Particles Swarm Optimization*.

1.3 Tujuan Penelitian

Tujuan penelitian yang ingin dicapai dalam tugas akhir ini adalah :

1. Menerapkan algoritma *Particles Swarm Optimization* dalam optimasi *Gate Assignment Problem*.
2. Meminimalkan *total walking distance* dalam permasalahan GAP.

1.4 Manfaat Penelitian

Manfaat yang ingin diperoleh dari penelitian tugas akhir ini bagi mahasiswa adalah untuk implementasi metode metaheuristik terutama algoritma *Particles Swarm Optimization* dalam kasus *Gate Assignment Problem*. Sedangkan manfaat yang sekiranya didapat oleh pihak terkait dari penelitian tugas akhir ini adalah mengetahui bahwa metode metaheuristik terutama algoritma *Particles Swarm Optimization* dapat digunakan dalam penyelesaian kasus *Gate Assignment Problem*.

1.5 Ruang Lingkup Penelitian

Berikut merupakan ruang lingkup dari penelitian tugas akhir ini yang terdiri dari batasan dan asumsi yang digunakan saat penelitian.

1.5.1 Batasan

Berikut merupakan batasan yang digunakan di dalam penelitian ini :

1. Data yang digunakan merupakan data yang bersumber dari Bandara Soekarno Hatta.
2. Terminal yang digunakan merupakan terminal 2D & 2E.
3. Penerbangan yang dipergunakan merupakan penerbangan internasional.
4. Sampel data berasal dari pengamatan selama 1 hari 1 malam (12.00 am sd 12.00 am).

1.5.2 Asumsi

Berikut merupakan asumsi yang digunakan di dalam penelitian ini :

1. Tidak ada perubahan regulasi bandara selama pengamatan dilangsungkan.
2. Tidak ada *flight delay* untuk setiap penerbangan.
3. Kapasitas ruang tunggu pada setiap *gate* mencukupi.
4. Kursi kosong yang ada di gate selama jeda waktu tidak diperhitungkan.
5. *Flight* yang mengalami *overlap* dalam sebuah *gate* akan dikenakan penalti.
6. Nilai TPWD yang diperhitungkan murni jarak penumpang berjalan kaki tidak ada unsur lain, baik penumpang berjalan kaki dari *arrival hall* menuju *gate*, dari *gate* menuju *departure hall* dan antara *gate* untuk penumpang transit.

1.6 Sistematika Penulisan

Laporan tugas akhir ini terdiri dari enam bab dengan sistematika penulisan sebagai berikut :

BAB I PENDAHULUAN

Pada bab pendahuluan dijelaskan mengenai hal-hal yang mendasari dilakukannya penelitian serta identifikasi masalah penelitian yang meliputi latar belakang, perumusan masalah, tujuan penelitian, manfaat penelitian, serta batasan dan asumsi yang digunakan dalam penelitian.

BAB II TINJAUAN PUSTAKA

Pada tinjauan pustaka berisi tentang uraian teori dari permasalahan dan metode yang digunakan yang diperoleh dari referensi yang akan digunakan sebagai landasan dalam kegiatan penelitian tugas akhir ini.

BAB III METODOLOGI PENELITIAN

Pada metodologi penelitian dijelaskan mengenai tahapan-tahapan yang dilakukan dalam penelitian.

BAB IV PEMBUATAN MODEL DAN ALGORITMA

Pada bab pembuatan model dan algoritma dijelaskan mengenai validasi model algoritma untuk menyelesaikan persoalan beserta langkah – langkah penyelesaian permasalahan dengan menggunakan model algoritma yang valid dan verifikasi model algoritma dalam penyelesaian permasalahan.

BAB V HASIL EKSPERIMEN DAN ANALISIS

Pada bab hasil eksperimen dan analisis dijelaskan hasil eksperimen dalam penyelesaian permasalahan serta analisis mengenai hasil eksperimen yang dilakukan.

BAB V KESIMPULAN DAN SARAN

Pada bab kesimpulan dan saran berisi tentang kesimpulan hasil penelitian dan saran untuk penelitian sejenis yang diharapkan selanjutnya.

(Halaman ini sengaja dikosongkan)

BAB 2
TINJAUAN PUSTAKA

Pada tinjauan pustaka berisi tentang uraian teori dari permasalahan dan metode yang digunakan yang diperoleh dari referensi yang akan digunakan sebagai landasan dalam kegiatan penelitian tugas akhir ini.

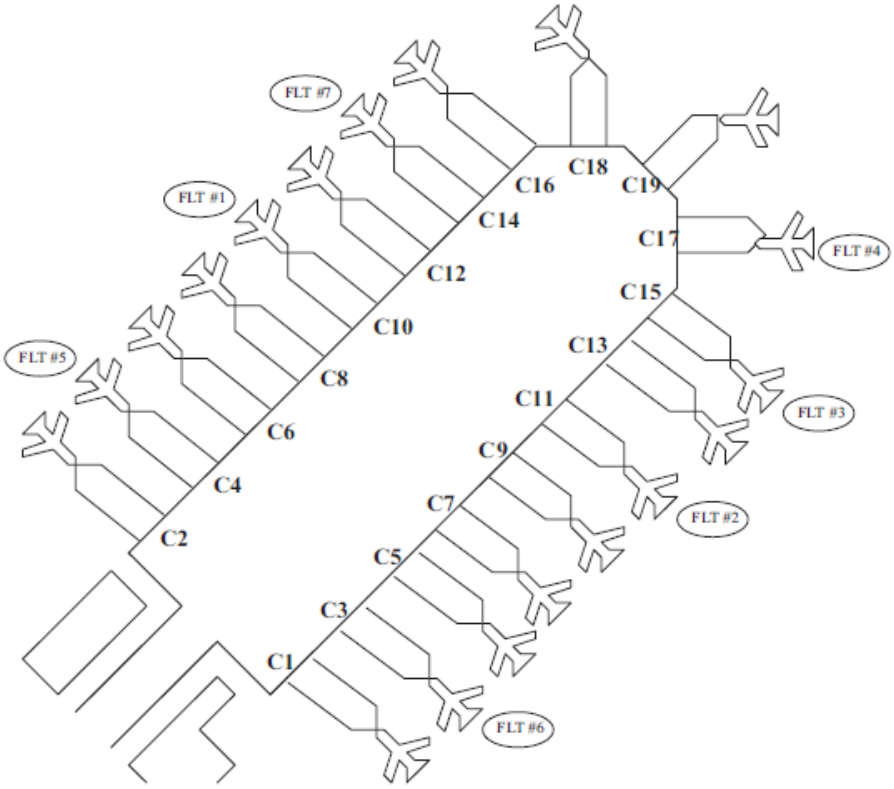
2.1 *Gate Assignment*

Gate Assignment bertujuan untuk menetapkan *flight* pada *gate* yang mampu meminimalkan ketidaknyamanan penumpang dan biaya operasional bandara dan maskapai penerbangan. Istilah *"gate"* digunakan tidak hanya untuk menunjuk fasilitas mengenai penumpang yang hendak *boarding* atau meninggalkan pesawat, tetapi juga posisi parkir setiap pesawat (Hu & Paolo, 2007). Terdapat beberapa faktor yang mempengaruhi *gate assignment* antara lain adalah ukuran pesawat, jarak penumpang berjalan, transfer bagasi, kemacetan jalan, rotasi pesawat, dan persyaratan layanan pesawat (Bazargan, 2010).

Permasalahan GAP dapat didefinisikan sebagai penetapan penugasan dari *flight* yang ada menuju *gate* yang tersedia dengan mengoptimalkan berbagai ukuran performansi yang menjadi batasan. Sebagian besar, GAP meminimalisasi total jarak penumpang berjalan kaki. Jarak yang ditempuh penumpang dapat dibedakan yakni jarak dari loket masuk (gerbang *check-in*) menuju *gate*, jarak dari *gate* menuju area bagasi (*baggage claim*) dan jarak dari satu *gate* menuju *gate* lainnya bagi penumpang transit (Aktel et al., 2016).

Permasalahan dalam menemukan *gate assignment* yang sesuai pada umumnya ditangai dalam tiga level. Pada level pertama, *ground controller* menggunakan jadwal penerbangan untuk memeriksa kapasitas *gate* untuk mengakomodasi penerbangan ini. Pada level kedua melibatkan pengembangan rencana harian sebelum hari operasi sebenarnya. Dan pada level ketiga, karena kondisi yang tidak dapat diatur seperti *delay*, cuaca buruk, kegagalan mesin, dan persyaratan *maintenance* maka rencana harian diperbarui dan direvisi pada jam/hari operasi yang sama (Bolat, 2000).

Hampir semua jenis pendekatan pemodelan diterapkan pada GAP dalam studi literatur. Teknik solusi yang berbeda juga diusulkan untuk memecahkan model. Terutama peneliti ingin mencari solusi yang optimal atau setidaknya berkualitas baik dalam waktu yang wajar. Metode solusi untuk GAP dapat diklasifikasikan sebagai metode eksak atau heuristik. Algoritma solusi eksak memastikan solusi optimal sedangkan algoritma heuristik tidak menjamin solusi optimal. Namun, studi terbaru kebanyakan berfokus pada heuristik untuk menyelesaikan GAP karena kompleksnya masalah (Aktel et al., 2016).



Gambar 2.1 Contoh *Gate Assignment* (Bazargan, 2010)

2.2 Optimasi Metaheuristik

Pencarian solusi dari sebuah model matematika dapat dilakukan dengan menggunakan dua metode yakni yang pertama adalah metode analitis yaitu dengan melakukan differensiasi atau gradien dari fungsi tujuan untuk mendapatkan solusi optimum, selanjutnya adalah metode numeris yaitu dengan melakukan percobaan pada beberapa nilai untuk menghasilkan solusi optimum (Munir, 2007). Metode numeris digunakan karena banyaknya permasalahan dalam kehidupan sehari-hari yang sulit untuk diselesaikan dengan menggunakan metode kalkulus ataupun analitik. Salah satu metode numeris yang sering digunakan dalam melakukan pendekatan adalah metode heuristik yang merupakan pendekatan komputasi untuk mencari solusi optimal atau mendekati optimal dari suatu problem optimasi secara iteratif untuk memperbaiki kandidat solusi dengan waktu yang cepat. Namun, metode heuristik biasanya hanya spesifik digunakan untuk problem tertentu yang membuat sebuah algoritma heuristik kurang fleksibel untuk digunakan, sehingga dikembangkan metode metaheuristik yang menggunakan algoritma yang bersifat umum dan menyelesaikan berbagai permasalahan (Santosa & Willy, 2011).

Metaheuristik adalah metoda untuk mencari solusi yang memadukan interaksi antara prosedur pencarian lokal dan strategi yang lebih tinggi untuk menciptakan proses yang mampu keluar dari titik-titik *local optimum* dan melakukan

pencarian di ruang solusi untuk menemukan solusi global. Algoritma metaheuristik biasanya digunakan untuk menyelesaikan permasalahan optimasi, baik pada permasalahan dengan variabel kontinyu maupun permasalahan dengan variabel diskrit. Contoh dari permasalahan optimasi dengan variabel kontinyu adalah pada penentuan dimensi produk untuk minimalisasi biaya suatu proyek. Sedangkan contoh permasalahan optimasi dengan variabel diskrit adalah penentuan rute transportasi. Pendekatan menggunakan algoritma metaheuristik memiliki karakteristik umum sebagai berikut (Santosa & Willy, 2011) :

1. Bersifat stokastik, menggunakan bilangan random untuk menentukan keputusan dalam salah satu langkah dalam algoritma..
2. Tidak menggunakan perhitungan gradien dari fungsi tujuan.
3. Biasanya diinspirasi dari analogi fisik, biologi, atau *ethology.*
4. Memiliki kelemahan dalam solusi, kesulitan mengatur nilai parameter dan komputasi yang lama, namun waktu komputasi juga kadang menjadi keunggulan disbanding optimasi eksak.

2.3 *Particles Swarm Optimization*

Particles Swarm Optimization (PSO) didasarkan pada perilaku sebuah kawanan burung atau ikan. Setiap individu atau partikel berperilaku dengan cara menggunakan kecerdasannya sendiri dan juga dipengaruhi perilaku kelompok kolektifnya. Dengan demikian, jika satu partikel atau seekor burung menemukan jalan yang tepat atau pendek menuju sumber makan, maka sisa kelompok yang lain juga akan dapat segera mengikuti jalan tersebut meskipun lokasi mereka jauh di kelompok tersebut. PSO telah diterapkan untuk menyelesaikan berbagai persoalan optimasi seperti VRP, penjadwalan proyek kerja dan beberapa masalah lainnya. Pada algoritma PSO ini, pencarian solusi dilakukan oleh suatu populasi yang terdiri dari beberapa partikel. Populasi dibangkitkan secara random dengan batasan nilai terkecil dan terbesar. Setiap partikel merepresentasikan posisi atau solusi dari permasalahan yang dihadapi. Setiap partikel merepresentasikan posisi atau solusi yang optimal dengan melintasi ruang pencarian. Hal ini dilakukan dengan cara setiap partikel melakukan penyesuaian terhadap posisi terbaik dari partikel tersebut (*local best*) dan penyesuaian terhadap posisi partikel terbaik dari seluruh kawanan (*global best*) selama melintasi ruang pencarian. Jadi, penyebaran pengalaman atau informasi terjadi di dalam partikel itu sendiri dan antara suatu partikel dengan partikel terbaik dari seluruh kawanan selama proses pencarian solusi. Setelah itu, dilakukan proses pencarian untuk mencari posisi terbaik setiap partikel dalam sejumlah iterasi tertentu sampai didapatkan posisi yang relatif *steady* atau mencapai batas iterasi yang ditetapkan. Pada setiap iterasi, setiap solusi yang direpresentasikan oleh posisi partikel, dievaluasi performansinya dengan cara memasukkan solusi tersebut ke dalam *fitness function* (Santosa & Willy, 2011).

Dalam beberapa tahun terakhir, PSO telah berhasil diterapkan pada banyak penelitian. Hal ini menunjukkan bahwa PSO mendapatkan hasil yang lebih baik

dengan cara yang lebih cepat dan lebih murah dibandingkan dengan metode lainnya. Alasan lain mengapa PSO menarik adalah bahwa terdapat beberapa parameter yang dapat disesuaikan. Satu versi, dengan sedikit variasi, bekerja dengan baik dalam berbagai macam aplikasi. PSO telah digunakan untuk pendekatan yang dapat digunakan di berbagai aplikasi, serta untuk aplikasi tertentu difokuskan pada persyaratan tertentu (Hu X. , 2006).

Berikut merupakan model matematika yang menggambarkan mekanisme *updating* status partikel :

$$V_i(t) = V_i(t-1) + c_1 r_1 \left(X_i^L - X_i(t-1)\right) + c_2 r_2 \left(X^G - X_i(t-1)\right) \quad (2.1)$$

$$X_i(t) = V_i(t) + V_i(t-1) \quad (2.2)$$

Dimana

$X_i^L = X_{i1}^L, X_{i2}^L, \dots, X_{iN}^L$ merepresentasikan *local best* dari partikel ke-i. Sedangkan $X_i^G = X_1^G, X_2^G, \dots, X_N^G$ merepresentasikan *global best* dari seluruh kawanan. Sedangkan c_1 dan c_2 adalah suatu konstanta yang bernilai positif yang biasanya disebut sebagai *learning factor*. Kemudian r_1 dan r_2 adalah suatu bilangan random yang bernilai antara 0 sampai 1. Persamaan 2.1 digunakan untuk menghitung kecepatan partikel yang baru berdasarkan kecepatang sebelumnya, jarak antara posisi saat ini dengan posisi terbaik partikel (*local best*), dan jarak antara posisi saat ini dengan posisi terbaik kawanan (*global best*). Kemudian partikel terbang menuju posisi yang baru berdasarkan persamaan 2.2. Lalu, algoritma PSO ini dijalankan dengan sejumlah iterasi tertentu hingga mencapai kriteria pemberhentian dan didapatkan solusi yang terletak pada *global best* (Santosa & Willy, 2011).

Berikut merupakan pseudo code algoritma PSO:

Inisialisasi partikel meliputi posisi awal dan kecepatan awal setiap partikel

Evaluasi fitness dari masing-masing partikel berdasarkan posisinya.

Pilihlah partikel dengan fitness terbaik dan tetapkan sebagai global best, sedangkan posisi awal sebagai local best.

Lakukan hingga iterasi maksimal atau minimal error criteria

Untuk setiap partikel

Perbarui kecepatan partikel berdasar global best dan local best

Perbarui posisi partikel berdasar kecepatan yang sudah diperbarui

Evaluasi fitness setiap partikel

Pilihlah partikel dengan fitness terbaik dan tetapkan sebagai global best

Tentukan local best dengan membandingkan posisi update dengan local best sebelumnya

2.4 *Literature Review*

Selain landasan teori yang telah dijelaskan pada subbab sebelumnya, penelitian ini juga didasari oleh penelitian-penelitian sebelumnya yang diambil dari

beberapa jurnal ilmiah dan tugas akhir. Penelitian-penelitian sebelumnya meliputi penelitian mengenai *Gate Assignment* dengan berbagai metode dan objektif yang berbeda-beda. Berikut ini adalah beberapa *critical review* yang berhubungan dengan penelitian ini.

Tabel 2.1 Literature Review Penelitian

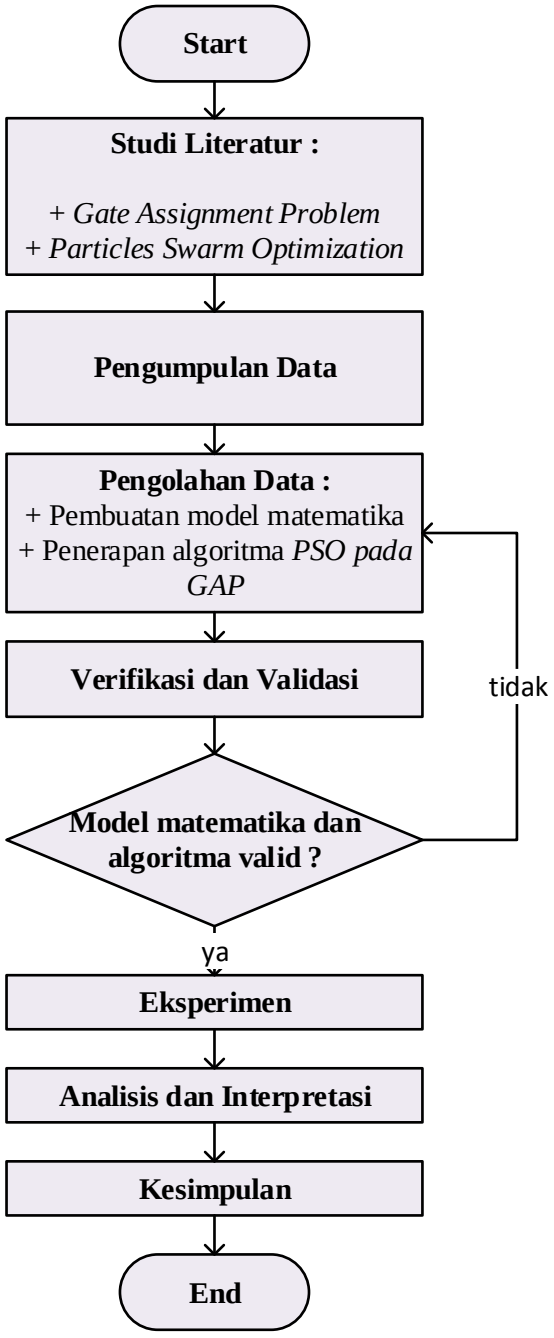
No	Judul Penelitian	Penulis	Metode	Tujuan
1	*A Metaheuristic Approach to Solve the Flight Gate Assignment Problem* (2015)	Mario Marinelli, Mauro Dell'Orco, Domenico Sassanelli	*Bee Colony Optimization*	Minimasi *total walking distasce* (TWD) dan minimasi jumlah *flight* yang diarahkan pada *remote terminal gates* (RG) berdasar maksimasi jumlah *flight* yang ditugaskan untuk *fixed gates* (FG)
2	*An Efficient Genetic Algorithm with Uniform Crossover for the Multi-Objective Airport Gate Assignment Problem* (2007)	Xiao-Bing Hu, Ezequiel Di Paolo	*Genetic Algorithm*	Minimasi *passenger walking distance, baggage transport distance, dan aircraft waiting time on the apron*
3	*The Airport Gate Assignment Problem – Multi Objective Optimization Versus Evoulutionati Multi Objective Optimization* (2017)	Ignacy Kaliszewaski, Janusz Miroforidisi, Jaroslaw Stanczak	*Mixed-Integer Programming (MIP) dan Evolutionary Multi-Objective Optimization (EMO)*	Membandingkan hasil dari 2 algoritma yang digunakan yakni MIP dan EMO

No	Judul Penelitian	Penulis	Metode	Tujuan
4	*The Comparison of The Metaheuristic Algorithms Performances on Airport Gate Assignment Problem* (2017)	Abdullah Aktel, Betul Yagmahan, Tuncay Ozcan, M. Mutlu Yenisey, Engin Sansarci	*Tabu Search* (TS), *Simulated Annealing* (SA) dan *Greedy-A*	Melakukam perbandingan performansi algoritma TS dan SA serta Greedy-A sebagai benchmark dalam AGAP
5	*Solving The Gate Assignment Problem through The Fuzzy Bee Colony Optimization* (2017)	Mauro Dell'Orco, Mario Marinelli, Maria Giovanna Altieri	*Bee Colony Optimization*	Minimasi *total walking distance* dan jumlah *flight* yang diarahkan pada *remote terminal gates*
6	*Fusion of Two Metaheuristic Approach to Solve The Flight Gate Assignment Problem* (2015)	Mario Marinelli, Gianvito Palmisano, Mauro Dell'Orco, Michele Ottomanelli	*Biogeography-based Bee Colony Optimization*	Minimasi *total walking distance* dan minimasi jumlah *flight* yang diarahkan pada *remote terminal gates*

Berdasar *literature review* pada Tabel 2.1 dapat dilihat bahwa penelitian-penelitian mengenai *Gate Assignment* sudah banyak diselesaikan dengan pendekatan metaheuristik. Metode metaheuristik memang cukup sesuai dengan kondisi permasalahan pada *Gate Assignment* dimana permasalahan yang ada cukup kompleks dengan mempertimbangan berbagai aspek seperti ukuran pesawat, ukuran *gate*, jadwal penerbangan, jarak penumpang berjalan dan sebagainya. Pada tugas akhir ini, metode yang akan digunakan untuk menyelesaikan permasalahan ini adalah metode metaheuristik *Particles Swarm Optimization* yang mana mempertimbangan seluruh yang aspek yang bersangkutan untuk meminimasi *total walking distance.*

BAB 3
METODOLOGI PENELITIAN

Pada metodologi penelitian dijelaskan mengenai tahapan-tahapan yang dilakukan dalam penelitian implementasi Algoritma *Particles Swarm Optimization* (PSO) pada *Gate Assignment Problem*. Berikut merupakan *flowchart* dari penelitian yang dilakukan.



Gambar 3.1 *Flowchart* Penelitian

3.1 Studi Literatur

Tahap studi literatur bertujuan untuk mencari referensi terkait dengan *Gate Assignment Problem* yang ada dalam dunia industri penerbangan terutama di bandara. Selain itu bertujuan pula untuk mencari referensi terkait dengan *algoritma PSO* yang digunakan dalam penelitian ini. Pencarian referensi juga dilakukan untuk mengetahui data-data yang diperlukan untuk menyelesaikan permasalahan ini.

3.2 Pengumpulan Data

Setelah diketahui permasalahan yang ada, dilakukan pengambilan data. Data yang dipergunakan yaitu (Rahmawati, 2014):

1. Data jumlah *flight* yang ada dalam terminal di bandara Soekarno-Hatta. Data ini digunakan untuk mengetahui banyaknya *flight* yang harus ditangani. Jumlah *flight* yang ada sebanyak 79.
2. Data jumlah *gate* yang ada dalam terminal di bandara Soekarno-Hatta. Data ini digunakan untuk menugaskan *gate* untuk setiap *flight*. Jumlah *gate* yang digunakan berjumlah 14.
3. Data jumlah penumpang yang datang dan jumlah penumpang yang berangkat dari setiap *flight*. Data ini didapatkan dari jumlah kursi yang tersedia pada setiap jenis pesawat yang digunakan dalam setiap *flight* dan dikalikan dengan bilangan antara *minimum requirement* untuk melalukan penerbangan sampai jumlah kursi terisi penuh.
4. Data jumlah transfer penumpang dari satu *flight* ke *flight* lain. Data berasal dari asumsi jumlah orang yang melakukan transfer penerbangan dan berpacu pada *connecting flight* pada waktu tertentu.
5. Data jarak transfer penumpang dari satu *flight* ke *flight* lain. Data ini didapatkan dari peta terminal 2 sub terminal 2D dan 2E pada bandara Soekarno-Hatta dengan skala 1:100 meter.
6. Data jarak penumpang berjalan dari setiap *gate* ke *arrival hall* dan *departure hall*. Digunakan sebagai salah satu parameter ukuran matriks jarak penumpang dalam berjalan. Data jarak setiap *gate* ke arrival hall dan departure hall didapatkan pula dari peta terminal 2 sub terminal 2D dan 2E pada bandara Soekarno-Hatta dengan skala 1:100 meter.

Data-data tersebut tidak diambil secara langsung melainkan dengan menggunakan data artifisial dimana data yang digunakan berasal dari sumber jurnal atau buku atau berdasarkan sumber-sumber yang ada. Data utama diambil dari data tugas akhir Siti Dwi Rahmawati berjudul Algoritma Modified Simulated Annealing Untuk Menyelesaikan Airport *Gate* Assignment Problem Studi Kasus Bandara Soekarno-Hatta tahun 2014.

3.3 Pengolahan Data

Data yang telah didapatkan dari proses pengambilan data kemudian diolah untuk menjadi model seperti berikut.

3.3.1 Pembuatan Model Matematis

Dalam tahap pengolahan data, langkah awal yang dilakukan yaitu membuat model matematis dari permasalahan yang akan diteliti. Berikut merupakan formulasi model matematis dalam penelitian ini.

Parameter :

f = *set flight*

g = *set gate*

a_i = waktu kedatangan dari *flight i*

d_i = waktu keberangkatan dari *flight i*

pt_{ij} = total jumlah *transfer passenger dari flight i ke flight j*

pa_i = total jumlah *arriving passenger dari flight i*

pd_i = total jumlah *departing passenger dari flight i*

wd_{mn} = jarak berjalan antara *gate m* dan *gate n*

wd_{0m} = jarak berjalan antara *gate m* dan *arrival hall*

wd_{m0} = jarak berjalan antara *departure hall* dan *gate n*

Decision Variable :

$$x_{ik} = \begin{cases} 1 & \textit{Jika flight I ditugaskan pada gate k} \\ 0 & \textit{otherwise} \end{cases}$$

Model Matematika :

$$Min \sum_{i\in f}\sum_{j\in f}\sum_{m\in G}\sum_{n\in G} wd_{mn}x_{ik}x_{im}x_{jn} + \sum_{i\in F}\sum_{m\in G}(pa_{im}\times wd_{0m} + pd_{im}\times wd_{m0})x_{im} \quad (3.1)$$

Subject to :

$$\sum_{m\in g} x_{mi} = 1\,, \qquad \forall m \in f \quad (3.2)$$

$$\sum_{i\in f} x_{im} = 1\,, \qquad \forall i \in f \quad (3.3)$$

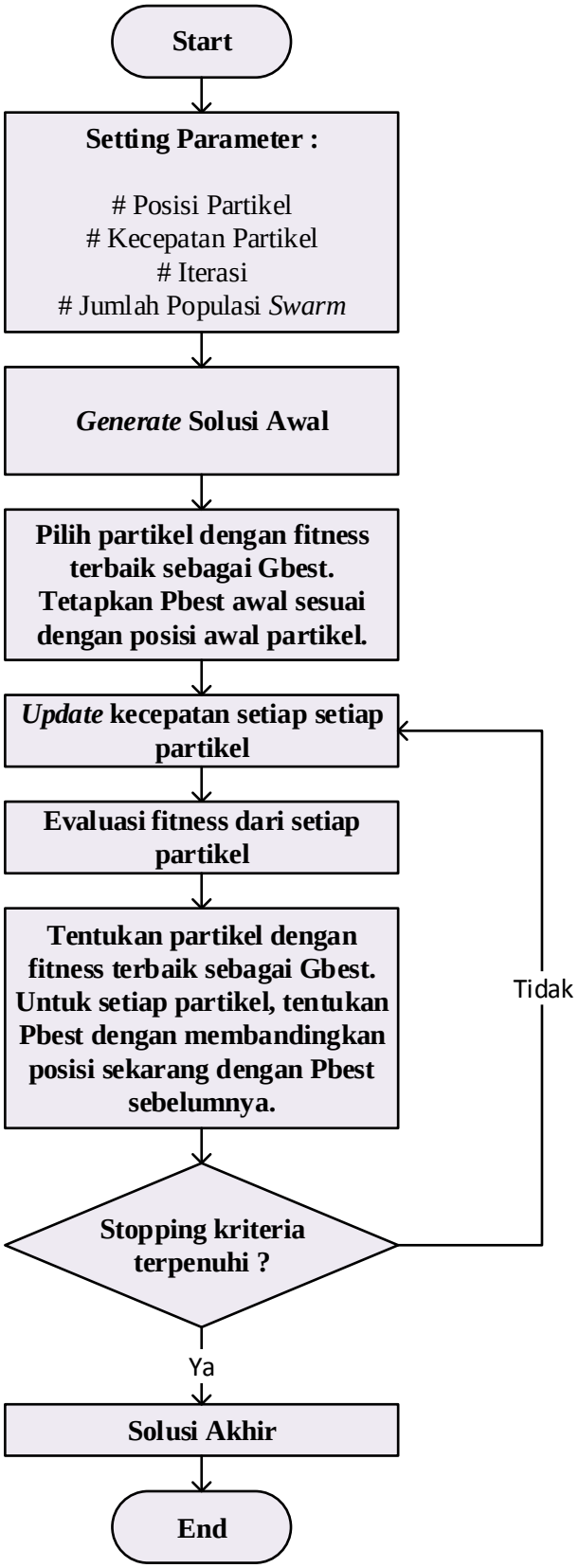
$$d_i < a_j \leftrightarrow x_{ij} = 1, \quad \forall i \in f\ \forall j \in f \quad (3.4)$$

Dalam model ini, fungsi tujuan (3.1) mencoba untuk meminimalkan total jarak penumpang berjalan. *Constraint* (3.2) menjamin bahwa setiap *gate* hanya ditugaskan untuk 1 *flights*. *Constraint* (3.3) memastikan bahwa setiap *flight* hanya ditugaskan

untuk 1 *gate*. *Constraint* (3.4) membahas bilamana waktu keberangkatan *flight i* lebih cepat daripada waktu kedatangan *flight j* maka *flight i* dan *j* dapat ditempatkan pada 1 *gate*.

3.3.2 Pembuatan Algoritma

Setelah model matematika dibuat, selanjutnya adalah pembuatan algoritma PSO untuk membantu menyelesaikan permasalahan GAP. Kode program yang dibuat untuk menerjemahkan algoritma dibuat menggunakan *software matlab*. Berikut merupakan *flowchart* algoritma PSO yang digunakan dalam penelitian ini.



Gambar 3.2 *Flowchart* Algoritma PSO yang dipergunakan

3.5 Verifikasi dan Validasi Model dan Algoritma

Setelah dilakukan pembuatan model dan penyusunan algoritma, maka langkah berikutnya adalah verifikasi dan validasi. Verifikasi adalah tahapan untuk mengetahui model matematika yang dibangun telah logis dan matematis serta data yang digunakan adalah tepat, sedangkan validasi merupakan tahapan untuk melihat apakah model yang telah dibuat mampu merepresentasikan permasalahan yang ada (Daellenbach & McNickle, 2005). Verifikasi dan validasi pada penelitian ini dilakukan menggunakan MATLAB dan Microsoft Excel.

3.6 Eksperimen dan Analisis

Setelah model dan algoritma diverifikasi dan divalidasi, maka selanjutnya dilakukan beberapa eksperimen untuk mencari hasil optimal dari permasalahan. Hal ini perlu dilakukan karena algoritma pada *software* membangkitkan bilangan *random* untuk inisialisasi solusi awal sehingga solusi bersifat deterministik, yaitu tidak selalu optimal. Hasil dari beberapa eksperimen ini selanjutnya perlu dianalisis sehingga dapat diketahui bagaimana hasil eksperimen serta mengetahui penyelesaikan yang didapatkan sesuai dengan permasalahan yang ada.

3.7 Penarikan Kesimpulan

Tahapan terakhir adalah penarikan kesimpulan dimana pada tahap ini merupakan jawaban dari tujuan dari penelitian ini dilakukan. Kesimpulan juga berdasarkan hasil analisis dan interpretasi dari hasil eksperimen yang sudah dilakukan. Selanjutnya, diberikan rekomendasi yang diharapkan mengenai pengembangan dari penelitian ini untuk penelitian selanjutnya.

BAB 4
PEMBUATAN MODEL DAN ALGORITMA

Pada pembuatan model dan algoritma dijelaskan mengenai tahapan pembuatan model *Gate* Assignment Problem (GAP) pada terminal 2 bandara internasional Soekarno – Hatta dan algoritma Particles Swarm Optimization untuk menyelesaikan permasalahan tersebut.

4.1 Model *Gate* Assignment Problem Terminal 2 Bandara Internasional Soekarno – Hatta

Model matematika dari *Gate* Assignment Problem (GAP) pada penelitian tugas akhir ini dibuat dari referensi berbagai sumber, yaitu penelitian-penelitian sebelumnya. Model yang dibuat pada tugas akhir ini dibuat berdasarkan referensi model yang telah dibuat dan diterapkan oleh (Rahmawati, 2014), (Aktel et all,2016) dan (Dell'Orco, Marinelli, & Altieri, 2017).

4.1.1 Pembuatan Model Matematis

Berikut merupakan notasi yang dipergunakan dalam pengerjaan penelitian ini, antara lain :

f = *set flight*

g = *set gate*

a_i = waktu kedatangan dari *flight i*

d_i = waktu keberangkatan dari *flight i*

pt_{ij} = total jumlah *transfer passenger dari flight i ke flight j*

pa_i = total jumlah *arriving passenger dari flight i*

pd_i = total jumlah *departing passenger dari flight i*

wd_{mn} = jarak berjalan antara *gate m* dan *gate n*

wd_{0m} = jarak berjalan antara *gate m* dan *arriving hall*

wd_{m0} = jarak berjalan antara *departure hall* dan *gate n*

4.1.2 Fungsi Tujuan

Dalam penelitian ini fungsi tujuan yang diterapkan sebagai pertimbangan utama dalam mencari solusi adalah meminimalkan *total passenger walking distance* dengan algoritma *Particles Swarm Optimization*.

$$Min \sum_{i\in f}\sum_{j\in f}\sum_{m\in G}\sum_{n\in G} wd_{mn}x_{ik}x_{im}x_{jn} + \sum_{i\in F}\sum_{m\in G}(pa_{im} \times wd_{0m} + pd_{im} \times wd_{m0})x_{im} \quad (4.1)$$

Berdasar fungsi tujuan tersebut didapati bahwa GAP ini bahasan utamanya terletak pada jarak berjalan penumpang yang dipengaruhi oleh penugasan *flight* pada *gate* yang baik. Penugasan *gate* yang baik tentunya dipengaruhi oleh ketersediaan *gate*, banyaknya penumpang dalam sebuah *flight*, jarak antara *hall* dengan *gate*, dan jarak antar *gate*.

4.1.3 Batasan

Berdasarkan model yang telah ada, dilakukan beberapa batasan dalam model matematika GAP ini. Dalam penelitian ini terdapat tiga batasan utama yaitu setiap *gate* hanya ditugaskan untuk satu *flight*, setiap *flight* hanya ditugaskan untuk satu *gate*, dan batasan waktu dimana tidak terjadi *flight* yang overlap pada setiap *gate*.

4.1.3.1 Batasan 1 : setiap gate hanya ditugaskan untuk 1 flight

$$\sum_{m\in g} x_{mi} = 1\,, \qquad \forall m \in f \qquad (4.2)$$

Dengan membatasi setiap *gate* hanya menerima 1 *flight* maka tidak akan membuat adanya 2 *flight* berada dalam 1 *gate* dalam waktu bersamaan. Dalam batasan ini setiap *gate* akan menerima *flight* yang berbeda. Berikut merupakan contoh solusi dari 8 *flight* dengan 3 *gate*.

Tabel 4.1 Contoh Solusi yang Tidak Melanggar Batasan

Gate	**1**	**2**	**3**
Flight	1	2	3
	6	4	5
	7	8	

Tabel 4.2 Contoh Solusi yang Melanggar Batasan

Gate	**1**	**2**	**3**
Flight	1	2	3
	6	4	5
	7	8	3

Pada Tabel 4.1 tidak ada duplikasi *flight* yang ditugaskan dalam setiap gate sehingga solusi tidak melanggar batasan yang ada. Sedangkan pada Tabel 4.2 terlihat solusi pada *gate* 3 terlihat ada duplikasi *flight* (3) sehingga batasan setiap *gate* hanya ditugaskan untuk 1 *flight* dilanggar.

4.1.3.2 *Batasan 2 : setiap flight hanya ditugaskan pada 1 gate*

$$\sum_{i \in f} x_{im} = 1\,, \qquad \forall i \in f \tag{4.3}$$

Dengan membatasi setiap *flight* hanya akan ditugaskan pada 1 *gate* tertentu maka *flight* lain tidak akan berada dalam *gate* yang sama dalam waktu yang yang bersamaan. Berikut merupakan contoh solusi dari 8 *flight* dengan 3 *gate*.

Tabel 4.3 Contoh Solusi yang Tidak Melanggar Batasan

Gate	**1**	**2**	**3**
Flight	1	2	3
	6	4	5
	7	8	

Tabel 4.4 Contoh Solusi yang Melanggar Batasan

Gate	**1**	**2**	**3**
Flight	1	2	3
	6	4	5
	7	8	8

Pada Tabel 4.3 tidak ada duplikasi *flight* yang ditugaskan dalam setiap gate sehingga solusi tidak melanggar batasan yang ada. Sedangkan pada Tabel 4.4 terlihat bahwa *flight (8)* ditugaskan pada *gate (2)* dan *gate (3)* sehingga batasan bahwa setiap *flight* hanya ditugaskan pada 1 *gate* dilanggar.

4.1.3.3 *Batasan 3 : setiap flight yang overlap tidak akan berada dalam gate yang sama*

$$d_i < a_j \leftrightarrow x_{ij} = 1, \quad \forall i \in f\,,\ \forall j \in f \tag{4.4}$$

Batasan ini membahas bilamana waktu keberangkatan *flight i* lebih cepat daripada waktu kedatangan *flight j* maka *flight i* dan *flight j* dapat ditempatkan pada satu *gate* yang sama. Sedangkan *flights* yang *overlap* tidak boleh ditempatkan pada satu gate yang sama. Istilah *overlap* yang dimaksud adalah dimana ketika *flight i* belum *take off* dan *flight j* sudah *landing*. Dengan demikian maka *flights* yang *overlap* tidak boleh ditempatkan pada satu *gate* yang sama. Untuk lebih jelas dapat dilihat contoh pada tabel berikut.

Tabel 4.5 Contoh Solusi yang Tidak Melanggar Batasan

Gate	**1**	**2**	**3**	***Gate***	**1**	**2**	**3**	***Gate***	**1**	**2**	**3**
Flight	1	2	3	***Arrival***	400	680	805	***Departure***	760	800	1005
	6	4	5		1090	975	1015		1170	1095	1190
	8	7			1200	1160	-		1380	1410	-

Tabel 4.6 Contoh Solusi yang Melanggar Batasan

Gate	**1**	**2**	**3**	***Gate***	**1**	**2**	**3**	***Gate***	**1**	**2**	**3**
Flight	1	2	3	***Arrival***	400	680	805	***Departure***	760	800	1005
	6	4	5		1090	975	1015		**1170**	1095	1190
	7	8			**1160**	1200	-		1410	1380	-

Pada Tabel 4.5 terlihat seluruh *arrival time flight* selanjutnya pada *flight* yang berada dalam satu gate tidak ada yang *overlap* dengan departure flight pada flight sebelumnya, seperti contoh pada *gate (1), arrival time flight (6)* terlihat pada menit ke-1090 sedangkan *departure time flight (1)* tertera pada menit ke-760 sehingga tidak ada overlap time disini. Sedangkan pada Tabel 4.6 dapat dilihat *arrival time flight (7)* tertera pada menit ke-1160, namun *departure time flight (6)* terlihat pada menit ke-1170 sehingga terjadi *overlap time* pada *gate (1)* yang mana melanggar batasan setiap *flight* yang *overlap* tidak akan berada pada *gate* yang sama.

4.2 Verifikasi dan Validasi Model Matematika

Pada subbab ini dibahas mengenai verifikasi model matematika yang dibuat apakah sesuai dengan dengan model matematika yang dicanangkan pada subbab sebelumnya dan validasi terkait model matematika yang dibuat dan algoritma pengerjaan model tersebut. Verifikasi adalah tahapan untuk mengetahui model yang dibangun telah tepat secara logis dan matematis, sedangkan validasi merupakan tahapan untuk melihat apakah model yang telah dibuat mampu merepresentasikan permasalahan yang diteliti. Untuk melakukan validasi diperlukan data kecil yang diperlukan untuk merepresentasikan permasalahan yang ada. Berikut digunakan kasus kecil dengan menggunakan delapan *flight* dan tiga *gate*.

Tabel 4.7 Waktu dan Jumlah Penumpang yang Tiba dan Berangkat

Flight	***Arrival***		***Departure***	
	time (at) **(menit)**	***passenger (ap)*** **(orang)**	***time (dt)*** **(menit)**	***passenger (dp)*** **(orang)**
1	400	130	760	140
2	680	104	800	112
3	805	125	1005	102
4	975	106	1095	110
5	1015	109	1085	113
6	1090	142	1170	138
7	1180	121	1410	111
8	1200	107	1380	123

Tabel 4.8 Matriks Jumlah Penumpang yang Melakukan *Transfer Flight*

Flight	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0	0	0	0	0	0	0	0
2	0	0	10	0	0	0	20	18
3	0	0	0	0	0	0	2	0
4	0	0	0	0	42	0	31	0
5	0	0	0	0	0	0	8	0
6	0	0	0	0	0	0	30	12
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

Tabel 4.9 Matriks Jarak Antar *Gate*

Gate	**1**	**2**	**3**
1	0	10	30
2	10	0	25
3	30	25	0

Tabel 4.10 Jarak *Gate* dengan *Arrival Hall* dan *Departure Hall*

Gate	**Jarak dengan *Arrival Hall* (meter)**	**Jarak dengan *Departure Hall* (meter)**
1	10	15
2	8	11
3	13	13

4.2.1 Verifikasi Model Matematika

Verifikasi model dilakukan dengan melakukan evaluasi struktur model yang di-*generate* dalam software *Microsoft Excel* untuk mengecek apakah model yang dibuat sudah sesuai dengan model matematisnya. Berikut merupakan struktur model yang dicanangkan dengan menggunakan *solver Microsoft Excel*.

Fungsi Tujuan:

Min = sum(walking distance transfer passenger)+sum(walking distance arrival passenger)+sum(walking distance departing passenger)

Variabel Pengubah :

Binary penugasan flight pada gate tertentu

$$x_{ik} = \begin{cases} 1 & \textit{Jika flight I ditugaskan pada gate k} \\ 0 & \textit{otherwise} \end{cases}$$

Batasan :

$Sum(x_{ik}) = 1$;

$Arriving\ time\ flight_{ik} < Arriving\ time\ flight_{jk}$;

$Departing\ time\ flight_{ik} < Departing\ time\ flight_{jk}$;

$Arriving\ time\ flight_{ik} < Departing\ time\ flight_{jk}$;

Berdasarkan model tersebut, dapat diketahui bahwa model yang di-*generate* sesuai dengan model matematis yang telah dibuat pada subbab 4.1 sehingga model matematika yang dibuat sudah terverifikasi. Setelah verifikasi dilakukan maka perlu untuk menguji apakah solusi yang dihasilkan sudah memenuhi batasan-batasan yang ada. Validasi dilakukan dengan membandingkan solusi dari *solver* dengan logika hasil perhitungan sehingga model dapat dikatakan layak. Solusi dari dari *solver* adalah sebagai berikut.

Tabel 4.11 *Output* Penugasan dengan *Solver*

Gate	**1**	**2**	**3**	**TPWD**
Flight	*2*	*3*	*1*	*23786*
	4	*6*	*5*	
	7	*8*	*0*	

Batasan 1 : setiap gate hanya ditugaskan untuk 1 flight

Terlihat bahwa tidak ada *flight* yang tertulis rangkap pada masing-masing *gate*. Dengan demikian batasan ini tidak dilanggar.

Batasan 2 : setiap flight hanya ditugaskan pada 1 gate

Terlihat bahwa tidak ada *flight* yang tertulis rangkap pada *gate* yang berbeda. Dengan demikian batasan ini tidak dilanggar.

Batasan 3 : setiap flight yang tumpang tidih waktunya tidak akan berada dalam gate yang sama

Tabel 4.12 Data *Arrival* dan *Departure Time* pada Masing-Masing *Gate*

Gate	**1**	**2**	**3**	***Gate***	**1**	**2**	**3**
Arrival	*680*	*805*	*400*	***Departure***	*800*	*1005*	*760*
	975	*1090*	*1015*		*1095*	*1170*	*1190*
	1160	*1200*	-		*1410*	*1380*	-

Terlihat bahwa pada solusi terlihat sebelum *flight* yang selanjutnya *landing* pada sebuah *gate*, *flight* yang sebelumnya sudah *take off* sehingga tidak terjadi *overlap*. Untuk itu batasan ini tidak dilanggar.

Pengujian batasan menunjukkan bahwa solusi yang dihasilkan dari *solver* tidak melanggar batasan yang ada. Nilai TPWD minimum yang didapatkan pada solver adalah 23786 meter. Dengan rincian penugasan sebagai berikut :

Gate 1 (2, 4, 7)

Gate 2 (3, 6, 8)

Gate 3 (1,5)

4.2.2 Langkah – Langkah Penyelesaian *Gate Assignment Problem* dengan Algoritma *Particles Swarm Optimization*

Particles Swarm Optimization atau yang kerap disebut PSO merupakan salah satu *population based algorithm* yang didasarkan pada perilaku sebuah kawanan burung atau ikan. Pada subbab ini berisi mengenai langkah – langkah yang digunakan dalam menyelesaikan GAP dengan menggunakan PSO.

Langkah 1 : Bangkitkan *Feasible Solution*

Berbeda dengan algoritma pada umumnya yang diawali dengan inisialisasi parameter, kali ini algoritma diawali dengan mendefinisikan batasan solusi yang dapat dibangkitkan. *Feasible solution* didapatkan dengan menerapkan batasan waktu yang menjadikan *flight* mana saja yang dapat dijadikan dalam satu *gate*. Sehingga nantinya solusi yang dibangkitkan mampu menyelesaikan permasalahan yang ada. Dalam kasus sederhana ini, didapatkan *flight* mana saja yang dapat ditempatkan pada *gate* yang sama sebagai berikut.

Tabel 4.13 Tabel *Flight* yang Dapat Ditempatkan pada *Gate* yang Sama

Flight	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	1	0	1	1	1	1	1	1
2	0	1	1	1	1	1	1	1
3	1	1	1	0	1	1	1	1
4	1	1	0	1	0	0	1	1
5	1	1	1	0	1	1	1	1
6	1	1	1	0	1	1	1	1
7	1	1	1	1	1	1	1	0
8	1	1	1	1	1	1	0	1

Pada Tabel 4.13 diketahui bahwa tidak semua *flight* dapat ditempatkan pada satu *gate* yang sama seperti *flight 1* dan *flight 2* tidak dapat ditempatkan pada *gate* yang sama. Hal ini sangat diperlukan sebagai basis bahwa solusi yang dibangkitkan harus memenuhi batasan mana saja *flight* yang dapat ditugaskan pada *gate* yang sama.

Langkah 2 : Inisialisasi Parameter dan Solusi Awal

Solusi awal dibangkitkan dengan memunculkan 1 populasi kawanan. Dalam tahap ini ditentukan pula jumlah iterasi yang diperlukan dalam menyelesaikan permasalahan. Dengan membangkitkan bilangan random sejumlah *flight* yang ada maka ditemukan alternative awal yang kemudian disesuaikan dengan *feasible solution* yang sudah ditetapkan sebelumnya. Dengan didapatkan solusi awal yang *feasible* maka dihitunglah fungsi tujuan awal yang menjadi dasar awal pengerjaan algoritma.

Tabel 4.14 Nilai Fungsi Tujuan Solusi Awal

Gate 1	*Gate* 2	*Gate* 3	TPWD
1	3	0	24784
4	6	2	
7	8	5	

Nilai Pbest dan Gbest diatur sesuai dengan posisi populasi pada penciptaan solusi awal. Nilai Pbest dan Gbest ini nantinya yang akan mempengaruhi kecepatan perpindahan posisi populasi untuk membangkitkan solusi baru.

Langkah 3 : Bangkitkan Solusi Baru

Seperti layaknya algoritma PSO pada umumnya dibangkitkan solusi baru dengan melakukan *update* populasi melalui kecepatan masing – masing partikel. Kecepatan partikel ditentukan oleh Gbest pada solusi awal. Posisi populasi awal ditambah dengan kecepatan yang mempengaruhi masing – masing partikel maka akan merubah posisi masing – masing partikel yang ada dalam populasi sehingga akan membuahkan posisi populasi yang baru yang nantinya dipergunakan untuk membuat solusi baru yang diharapkan mampu lebih baik dari solusi sebelumnya.

Tabel 4.15 Nilai Fungsi Tujuan Solusi Baru

Gate 1	*Gate* 2	*Gate* 3	TPWD
2	1	0	23889
6	4	3	
8	7	5	

Solusi baru akan dijadikan pembandingan dengan solusi sebelumnya. Apabila solusi yang didapatkan lebih baik maka solusi baru baru akan menggantikan solusi awal. Demikian pula sebaliknya apabila solusi awal lebih baik maka solusi awal akan tetap menjadi solusi akhir.

Pada Tabel 4.14 dan Tabel 4.15 didapatkan bahwa nilai TPWD solusi baru lebih rendah dibanding solusi awal sehingga solusi baru menggantikan solusi awal. Dalam hal ini nilai Gbest yang merupakan posisi terbaik populasi akan dilakukan *update* berdasar posisi populasi baru.

Langkah 4 : Output Algoritma Particles Swarm Optimization

Setelah melalui berbagai proses tersebut maka didapatilah output yang diperlukan dalam penyelesaian GAP ini. Pada akhirnya diperoleh solusi akhir *Total Passenger Walking Distance* (TPWD) yang menjadi tujuan utama dalam permasalahan ini disertai dengan durasi lamanya algoritma dikerjakan dalam sekali jalan dengan sejumlah iterasi tertentu.

Berikut merupakan hasil output TPWD yang diperoleh dengan melalui 1000 iterasi menggunakan algoritma tersebut :

```
>>
[tpwds,time,GAsolve]=gappso2(fe,ge,ate,dte,ape,dpe,tpe,wdge,wdgae,wdgde,1000)

tpwds =

      23786

time =

   0.8594

GAsolve =

    2    3    0

    4    6    1

    7    8    5
```

Didapatkan hasil nilai TPWD sebesar 23786 meter. Dengan rincian penugasan sebagai berikut :

Gate 1 (2, 4, 7)

Gate 2 (3, 6, 8)

Gate 3 (1,5)

4.3 Verifikasi dan Validasi Algoritma

Pada bagian ini dilakukan verifikasi kode program yaitu dengan melakukan pengecekan pada model matematika yang sudah dibuat sebelumnya dan melihat ada atau tidaknya *error* pada kode program yang dibuat. Setiap langkah algoritma dilakukan pengecekan apakah sudah sesuai dengan yang diharapkan atau tidak. Saat kode program sudah berjalan sebagaimana mestinya dan *output* yang dihasilkan seperti dengan yang diharapkan. *Output* dengan menggunakan SOLVER dan MATLAB menunjukkan hasil yang sama maka dapat dikatakan bahwa kode program sudah terverifikasi dan tervalidasi.

BAB 5

EKSPERIMEN DAN ANALISIS

Pada bab ini berisi mengenai hasil eksperimen yang telah dilaksanakan beserta analisis hasil uji parameter algoritma *Particles Swarm Optimization* (PSO) dalam menyelesaikan *Gate Assignment Problem* (GAP) di bandara internasional Soekarno – Hatta.

5.1 Deskripsi Data

Data yang digunakan untuk menyelesaikan GAP merupakan data yang berasal dari terminal 2 bandara internasional Soekarno – Hatta. Data yang digunakan terdiri dari berbagai macam, seperti berikut :

1. Data jumlah *flight* yang ada dalam terminal di bandara Soekarno-Hatta.

   Data diperoleh dengan melakukan pengamatan secara langsung pada terminal 2 bandara Soekarno – Hatta. Data ini digunakan untuk mengetahui banyaknya *flight* yang harus ditangani. Jumlah *flight* yang ada sebanyak 79.

2. Data jumlah *gate* yang ada dalam terminal di bandara Soekarno-Hatta.

   Data jumlah *gate* yang ada dalam terminal di bandara Soekarno-Hatta. Data ini digunakan untuk menugaskan *gate bagi* setiap *flight*. Jumlah *gate* yang digunakan berjumlah 14.

3. Data waktu kedatangan dan keberangkatan setiap *flight.*

   Data diperoleh dari *flight* scheduling yang ada pada website PT Angkasa Pura II yang mengelola bandara Soekarno – Hatta. Data ini akan dijadikan acuan mengenai *flight* mana saja yang dapat ditempatkan pada *gate* yang sama.

4. Data jumlah penumpang yang tiba dan berangkat setiap *flight*

   Data jumlah penumpang yang datang dan jumlah penumpang yang berangkat dari setiap *flight*. Data ini didapatkan dari jumlah kursi yang tersedia pada setiap jenis pesawat yang digunakan dalam setiap *flight* dan dikalikan dengan bilangan antara *minimum requirement* untuk melalukan penerbangan sampai jumlah kursi terisi penuh.

5. Data jumlah transfer penumpang dari satu *flight* ke *flight* lain

   Data jumlah transfer penumpang dari satu *flight* ke *flight* lain. Data berasal dari asumsi jumlah orang yang melakukan transfer penerbangan dan berpacu pada *connecting flight* pada waktu tertentu.

6. Data jarak transfer penumpang dari satu *flight* ke *flight* lain.

   Data jarak transfer penumpang dari satu *flight* ke *flight* lain. Data ini didapatkan dari peta terminal 2 sub terminal 2D dan 2E pada bandara Soekarno-Hatta dengan skala 1:100 meter.

7. Data jarak penumpang berjalan dari setiap *gate* ke *arrival hall* dan *departure hall*.

   Data jarak penumpang berjalan dari setiap *gate* ke *arrival hall* dan *departure hall*. Digunakan sebagai salah satu parameter ukuran matriks jarak penumpang dalam berjalan. Data jarak setiap *gate* ke arrival hall dan departure hall didapatkan pula dari peta terminal 2 sub terminal 2D dan 2E pada bandara Soekarno-Hatta dengan skala 1:100 meter.

5.2 Hasil Eksperimen

Parameter yang dipergunakan dalam algoritma PSO untuk menyelesaikan kasus GAP ini adalah jumlah iterasi yang digunakan. Dimana populasi yang digunakan hanya 1 populasi. Dalam setiap iterasi nantinya didapatkan Gbest yang merupakan posisi populasi terbaik dalam penuntasan kasus GAP, dimana posisi terbaik dalam sejumlah iterasi tertentu akan menunjukkan nilai optimal dari solusi kasus GAP ini. Nilai fungsi tujuan dalam GAP ini adalah *total passenger walking distance* (TPWD) yang baik. Jumlah iterasi yang dipergunakan dalam penyelesaian kasus GAP ini adalah sejumlah 100 dan 1000 iterasi serta dilakukan replikasi sebanyak 5 kali. Berikut merupakan hasil replikasi yang didapatkan.

Table 5.1 Tabel Hasil *Running* Matlab dengan 100 iterasi

Iterasi = 100																
Replikasi ke	***Gate***														**TPWD**	**Waktu Komputasi**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**		
1	12	4	1	8	5	11	6	10	7	0	2	0	0	13	214010	0.6563
	24	28	15	22	17	23	18	19	16	9	26	0	3	21		
	40	41	31	25	39	29	33	34	32	20	36	0	27	30		
	52	51	47	37	49	44	50	46	54	35	45	14	42	43		
	61	70	53	55	60	58	59	69	65	48	66	38	57	64		
	75	77	62	63	79	74	68	73	72	67	76	56	71	78		
2	4	7	11	12	2	3	8	0	0	13	0	0	0	5	230135	0.6563
	26	19	18	25	17	20	22	6	14	24	9	10	1	28		
	32	31	29	35	39	34	30	21	27	38	16	23	15	41		
	56	47	44	49	46	53	50	33	42	45	37	36	40	54		
	68	58	64	60	52	59	57	51	65	67	55	43	48	70		
	77	79	78	75	63	73	71	61	72	76	66	62	69	74		
3	4	1	12	6	0	2	10	13	0	0	0	0	3	5	230140	0.6563
	18	15	19	16	8	20	25	27	11	9	14	7	17	21		
	33	32	31	30	23	29	39	41	28	24	26	22	35	34		
	47	46	45	56	37	43	44	49	42	38	40	36	54	48		
	58	57	67	64	51	65	53	63	70	52	55	50	66	68		
	78	77	75	72	61	74	62	71	76	60	69	59	73	79		

Tabel 5.1 Tabel Hasil *Running* Matlab dengan 100 iterasi (Lanjutan)

Iterasi = 100																
Replikasi ke	***Gate***														**TPWD**	**Waktu Komputasi**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**		
4	10	0	0	2	5	11	6	12	13	0	0	7	0	3	214005	0.625
	25	4	9	15	17	26	16	20	23	1	14	27	8	22		
	34	19	21	40	33	31	39	35	38	18	24	41	28	30		
	53	36	32	46	47	45	56	55	50	29	37	48	42	44		
	62	51	49	58	52	69	68	59	63	43	54	66	60	57		
	74	64	65	75	70	79	76	72	78	67	61	77	73	71		
5	8	1	4	6	3	2	10	11	7	5	0	0	0	0	215092	0.6406
	18	21	17	27	23	15	24	19	16	22	13	0	12	9		
	32	35	40	41	33	37	39	38	31	36	26	14	25	20		
	47	53	48	46	51	52	49	54	55	45	34	28	30	29		
	57	61	66	58	63	59	56	60	62	67	50	42	44	43		
	79	76	77	72	73	71	65	75	74	78	69	70	68	64		

Tabel 5.2 Tabel Hasil *Running* Matlab dengan 1000 iterasi

Iterasi = 1000																
Replikasi ke	***Gate***														**TPWD**	**Waktu Komputasi**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**		
1	9	5	4	13	11	7	8	0	0	2	0	0	0	3	213940	6.2188
	19	21	27	24	22	23	25	1	10	15	6	12	14	26		
	34	39	41	37	30	38	36	18	20	33	16	17	28	32		
	51	52	53	50	44	49	48	40	31	55	29	35	42	46		
	68	64	59	60	63	57	70	54	47	65	43	45	58	56		
	76	72	77	79	71	73	78	62	61	75	66	67	74	69		
2	1	13	9	8	14	4	5	2	0	0	0	0	0	3	214990	6.1719
	16	25	23	18	28	19	20	17	12	6	10	7	11	15		
	40	37	35	29	41	31	32	36	26	21	24	22	27	30		
	52	50	49	44	51	45	46	53	39	33	38	34	42	43		
	60	65	56	59	61	66	57	63	55	47	54	48	62	64		
	74	78	68	73	75	77	79	71	70	58	69	67	76	72		
3	12	9	11	0	5	10	7	2	14	0	0	0	0	1	213762	6.4844
	18	24	19	8	25	23	20	26	28	4	6	3	13	15		
	37	38	36	21	34	32	31	39	41	17	22	16	27	30		
	49	50	47	40	48	46	52	56	51	33	29	35	42	44		
	64	57	59	53	67	54	65	69	66	55	43	45	58	62		
	73	63	72	60	79	70	74	77	76	61	68	71	75	78		

Tabel 5.2 Tabel Hasil *Running* Matlab dengan 1000 iterasi (Lanjutan)

Iterasi = 1000																
Replikasi ke	***Gate***														**TPWD**	**Waktu Komputasi**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**		
4	1	4	9	6	8	2	11	3	5	0	0	0	0	14	214945	6.2344
	25	26	22	20	17	23	21	19	15	7	0	13	10	27		
	39	36	29	33	35	37	32	38	30	18	12	16	24	41		
	48	52	43	56	47	53	45	46	44	31	28	34	40	49		
	68	64	65	66	57	62	70	58	50	55	42	51	54	63		
	79	72	74	75	71	73	78	77	67	60	61	69	59	76		
5	11	5	7	6	0	1	12	0	3	0	14	0	0	10	213390	6.1875
	18	15	16	25	9	21	23	4	27	2	24	13	8	22		
	37	39	38	35	20	32	40	19	41	26	36	28	17	33		
	52	51	56	46	31	48	54	29	47	34	45	42	30	53		
	64	63	70	55	49	68	60	44	59	50	69	61	43	62		
	72	71	74	65	67	75	73	58	79	57	78	76	66	77		

Dari hasil running dengan 100 iterasi dan 1000 iterasi algoritma PSO dengan *software* matlab maka diperoleh TPWD masing – masing replikasi. Di bawah ini merupakan tabel perbandingan hasil replikasi yang didapatkan.

Tabel 5.3 Perbandingan Nilai TPWD

Replikasi ke	TPWD 100 iterasi	TPWD 1000 iterasi
1	214010	213940
2	230135	214990
3	230140	213762
4	214005	214945
5	215092	213390

Berdasarkan hasil 5 replikasi yang diperoleh didapatkan bahwa nilai TPWD yang diperoleh kala algoritma dijalankan dengan 100 iterasi masih diperoleh beberapa solusi yang kurang baik yakni 230135 dan 230140. Nilai TPWD minimum dengan 100 iterasi diperoleh 214005. Sedangkan kala iterasi ditambah hingga 1000 iterasi semua solusi sudah terlihat *feasible*, meskipun ada beberapa solusi yang terlihat tidak lebih baik kala hanya 100 iterasi yang dijalankan. Nilai TPWD minimum yang diperoleh kala 1000 iterasi dijalankan yaitu 213390.

Berikut merupakan hasil replikasi terbaik yang didapatkan dimana nilai TPWD minimum yang didapatkan sebesar 213390 dalam skala 1:100 meter atau setara 21339 km. Dengan penugasan :

Gate 1 (11, 18, 37, 52, 64, 72)

Gate 2 (5, 15, 39, 51, 63, 71)

Gate 3 (7, 16, 38, 56, 70, 74)

Gate 4 (6, 25, 35, 46, 55, 65)

Gate 5 (0, 9, 20, 31, 49, 67)

Gate 6 (1, 21, 32, 48, 68, 75)

Gate 7 (12, 23, 40, 54, 60, 73)

Gate 8 (0, 4, 19, 29, 44, 58)

Gate 9 (3, 27, 41, 47, 59, 79)

Gate 10 (0, 2, 26, 34, 50, 57)

Gate 11 (14, 24, 36, 45, 69, 78)

Gate 12 (0, 13, 28, 42, 61, 76)

Gate 13 (0, 8, 17, 30, 43, 66)

Gate 14 (10, 22, 33, 53, 62,77)

5.3 Analisis Performansi Algoritma *Particles Swarm Optimization*

Dengan menggunakan satu populasi tingkat performasi algoritma PSO praktis hanya bertumpu pada jumlah iterasi yang dibebankan kala melakukan *running* solusi. Semakin banyak iterasi yang dibebankan kala melakukan *running* menunjukkan semakin baik solusi yang didapatkan. Pada kasus ini dilakukan *running*

solusi sebanyak 100 dan 1000 iterasi. Terlihat bahwa pada 1000 iterasi solusi yang didapatkan secara normatif lebih stabil daripada solusi pada 100 iterasi. Hal ini dikarenakan probabilitas mendapatkan solusi yang lebih baik akan lebih terbuka pada iterasi yang besar. Dimana dengan semakin banyak iterasi maka ruang pencarian solusi akan lebih besar. Selain itu, dengan diberinya sebuah algoritma yang membuat pergerakan partikel dalam populasi akan semakin kecil seiring bertambahnya iterasi, sehingga apabila didapat solusi yang lebih baik maka semakin kecil perpindahan partikel dalam populasi tersebut. Dengan semakin kecilnya perpindahan partikel dalam populasi maka perubahan posisi partikel akan semakin kecil pula sehingga solusi yang baik mampu diraih dari iterasi ke iterasi.

Dari hasil *running* eksperimen yang didapatkan masih dapat dikatakan cukup baik bila dilihat dari waktu komputasi dimana dengan jumlah iterasi yang tidak sedikit waktu komputasi berjalan cukup cepat. Pada kasus ini ketika dijalankan 1000 iterasi hanya membutuhkan waktu sekitar 6 detik. Pada umumnya semakin besar iterasi akan memberikan *trade off* bahwa waktu komputasi akan semakin lama meskipun jumlah iterasi yang besar dan waktu komputasi yang lama tidak menjamin akan didapatkannya solusi yang baik. Waktu komputasi kala menjalankan algoritma yang diprogram di MATLAB terlihat cepat dapat diakibatkan oleh sedikitnya data penumpang transfer.

Selain itu pula dengan diberinya algoritma supaya pergerakan partikel akan semakin kecil kala mendapati solusi yang lebih baik membuat waktu *looping* akan semakin berkurang. Waktu komputasi yang cepat dapat pula diakibatkan karena tidak diperhitungkannya kapasitas *seat* yang ada pada setiap *gate*, sehingga setiap *gate* diasumsikan memiliki kapasitas yang mencukupi. Dengan ini, setiap *flight* memiliki bobot yang sama karena dapat ditempatkan di semua *gate*, hanya di dalam algoritma diberikan *trade off* grup *flight* yang terdapat dalam satu *gate* yang memiliki jumlah penumpang terbanyak akan ditempatkan pada *gate* yang memiliki jarak terdekat terhadap *arrival* dan *departure hall*.

5.4 Analisis Hasil Eksperimen

Penggunaan satu populasi sangatlah terlihat dalam hasil replikasi yang didapatkan, dimana untuk mencapai solusi optimal membutuhkan iterasi yang sangatlah besar. Namun, seperti yang diketahui bahwa performa PSO tidak menjamin semakin banyak iterasi akan menjamin didapatkan hasil yang optimal. Dalam hasil replikasi yang didapatkan diketahui bahwa solusi masih dapat dioptimalkan kembali sehingga hasil eksperimen meskipun belum optimal tetap memberikan representatif solusi dari permasalahan yang ada. Hal ini dapat dipecahkan dengan memberikan penambahan iterasi yang jauh berlipat sehingga harus memakan durasi *running* yang lebih lama namun solusi yang didapat lebih baik.

Terdapat beberapa replikasi meskipun jumlah iterasi yang dibebankan dinaikkan 10 kali lipat tetap tidak memberikan solusi yang lebih baik. Nilai TPWD hasil replikasi dengan 100 dan 1000 iterasi tidaklah berbeda sangat jauh. Hal ini dapat diakibatkan bahwa posisi populasi pada algoritma PSO berada pada posisi yang kurang baik kala diproses untuk didapatkan solusi minimum TPWD pada GAP.

Berdasar hasil terbaik dari 5 replikasi yang didapatkan nilai TPWD sebesar 213390 dalam skala 1:100 meter atau setara 21339 km. Nilai TPWD ini seharusnya masih dapat lebih dioptimalkan dengan mengurangi berbagai alternatif. Seperti, mengurangi jumlah *flight* pada *gate* yang memiliki jarak terjauh dengan *arrival* dan *departure hall*. Dalam kasus ini terdapat empat *gate* yang memiliki jarak terjauh dengan *arrival* dan *departure hall* yakni *gate (10)*, *gate (11)*, *gate (12)*, dan *gate (13)*. Pemberian beban *flight* yang seminimal mungkin pada ketiga *gate* ini maka akan memberikan *trade off gate* yang lain mendapat *flight* tambahan, namun dengan demikian nilai TPWD dimungkinkan berkurang karena jarak penumpang berjalan kaki semakin berkurang.

Peningkatan hasil TPWD dapat pula dilakukan dengan menaruh *flight* yang memiliki *transfer passenger* terbanyak ditempatkan pada *gate* yang berdampingan terlebih dapat ditempatkan pada *gate* yang sama. Dengan menaruh *flight* yang memiliki *transfer passenger* yang besar untuk ditugaskan pada *gate* yang sama tentunya membuat jarak penumpang transit berjalan kaki menjadi tidak ada. Sedangkan dengan menaruh *flight* yang memiliki *transfer passenger* yang besar untuk ditugaskan pada *gate* yang berdampingan maka jarak penumpang transit berjalan kaki menjadi dekat. Dengan ini sangat memungkinkan nilai TPWD akan mampu berkurang.

Pilihan lain yang dapat dilakukan untuk meminimalkan nilai TPWD adalah dengan melakukan penumpukan *flight* yang memiliki jumlah penumpang terbanyak untuk ditugaskan pada satu *gate*. Dengan menaruh *flights* yang memiliki jumlah penumpang besar pada satu *gate* akan membuat beban pada beberapa gate menjadi tidak seimbang dengan *gate* yang lain. Namun, dengan ini akan memberikan sebuah *trade off* dimana jarak penumpang berjalan kaki menjadi lebih dekat terhadap *arrival* dan *departure gate*. Dengan memberikan beban berlebih pada beberapa gate yang memiliki jarak terdekat dengan *arrival* dan *departure hall*, akan memberikan padatnya aktivitas pada *gate* tersebut. Padatnya aktivitas yang terjadi akan selaras dengan berkurangnya nilai TPWD yang dihasilkan.

(Halaman ini sengaja dikosongkan)

BAB 6

KESIMPULAN DAN SARAN

Pada bab ini berisi mengenai kesimpulan dari hasil penelitian yang telah dilakukan dan saran-saran untuk penelitian sejenis selanjutnya.

6.1 Kesimpulan

Berdasar hasil eksperimen dan analisi yang telah dilakukan dalam ditarik kesimpulan sebagai berikut :

1. Algoritma Particles Swarm Optimization (PSO) dapat menyelesaikan *Gate* Assignment Problem (GAP) dengan didapatkan minimum total *passenger walking distance* (TPWD) sebesar 21339 kilometer dan seluruh *flight* mampu ditugaskan pada *gate* yang tersedia. Hasil yang didapat ini memang bukanlah solusi terendah TPWD pada kasus ini, namun sudah cukup memberikan bukti bahwa dalam waktu relatif singkat PSO dapat digunakan untuk menyelesaikan kasus GAP.
2. Semakin besar iterasi yang diberikan pada algoritma PSO akan memberikan solusi yang lebih optimal dalam menyelesaikan GAP, dimana dengan 1000 iterasi solusi yang didapatkan lebih baik daripada 100 iterasi.
3. Waktu komputasi yang diperlukan kala menggunakan algoritma PSO dalam menyelesaikan GAP dapat dikatakan cukup singkat meskipun jumlah iterasi yang dibebankan cukup besar dimana pada 1000 iterasi hanya memakan waktu sekitar 6 detik.

6.1 Saran

Saran yang dapat diberikan berdasar pelaksanaan eksperimen untuk keperluan penelitian sejenis selanjutnya adalah sebagai berikut :

1. Peneliti perlu untuk benar – benar memperhatikan batasan – batasan yang dibebankan dalam menciptakan solusi yang *feasible* untuk kasus GAP.
2. Modifikasi algoritma yang lebih baik akan mampu memberikan solusi yang lebih optimal.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

Aktel, A., Yagmahan, B., Ozcan, T., Yenisey, M. M., & Sansarci, E. (2016). T*he Comparison of The Metaheuristic Algorithms Performances on Airport Gate Assignment Problem. Transportation Research Procedia 22*, 469 - 478.

Bazargan, M. (2010). *Airline Operations and Scheduling* (2nd ed.). Farnham: Ash*gate*.

Bolat, A. (2000). *Procedures for Providing Robust Gate Assignment for Arriving Aircrafts. European Journal of Operational Research*, 63-80.

Dell'Orco, M., Marinelli, M., & Altieri, M. G. (2017). Solving The Gate Assignment Problem Through The Fuzzy Bee Colony Optimization.

Deng, W., Zhaoa, H., Yang, X., Xiong, J., Sun, M., & Li, B. (2017). Study on An Improved Adaptive PSO Algorithm for Solving Multi Objective *Gate* Assignment.

Hu, X. (2006). Retrieved from Particle Swarm Optimization: http://www.swarmintelligence.org/

Hu, X.-B., & Paolo, E. D. (2007). *An Efficient Genetic Algorithm with Uniform Crossover for Multi-Objective Airport Gate Assignment Problem.*

Marinelli, M., Palmisano, G., Dell'Orco, M., & Ottomanelli, M. (2015). *Fusion of Two Metaheuristic Approaches to Solve The Flight Gate Assignment Problem. 18th Euro Working Group on Transportation, EWGT 2015*.

Miroforidis, J., Stańczak, J. T., & Kaliszewski, I. (2017). *The Airport Gate Assignment Problem - Multi Objective Optimization versus Evolutionary*.

Munir, R. (2007). Metode Analisa Numerik. Bandung: Penerbit Informatika.

Rahmawati, S. D. (2014). *Algoritma Modified Simulated Annealing Untuk Menyelesaikan Airport Gate Assignment Problem Studi Kasus Bandara Soekarno-Hatta.*

Santosa, B., & Willy, P. (2011). Metoda Metaheuristik Konsep dan Implementasi. Surabaya: Guna Widya.

Wittmer, A., Bieger, T., & Muller, R. (2011). *Aviation Systems.* Berlin: Springer.

LAMPIRAN 1

KODE MATLAB ALGORITMA PARTICLES SWARM OPTIMIZATION UNTUK MENYELESAIKAN *GATE* ASSIGNMENT PROBLEM

```
function
[tpwds,time,GAsolve]=gappso2(f,g,at,dt,ap,dp,tp,wdg,wdga,wdgd,itera
si)
%UNTUK MENDAPATKAN MINIMUM TOTAL PASSENGER WALKING DISTANCE (TPWD)
dengan 1
%populasi PSO
%f = set aircraft
%g = set gate
%at = arrival time
%dt = departure time
%tp = total transfer passenger
%ap = total arriving passenger
%dp = total departing passenger
%wdg = walking distance between gate
%wdga = walking distance between gate and arrival hall
%wdgd = walking distance between gate and departure hall
%it = iterasi
starttime=cputime;

%Generate Feasible Flights yang berada dalam 1 gate yang sama
A=mod(f,g);
if A==0
x=zeros(f);
else x=zeros(f+1);
for i=1:f
    for j=1:f
        if  dt(i)<at(j)
            x(i,j)=1;
        elseif at(i)>dt(j)
            x(i,j)=1;
        elseif i-j==0
            x(i,j)=1;
        else x(i,j)=0;
        end
    end
end
end
x(:,(f+1))=1;x((f+1),:)=1;%untuk fake flight jika jumlah kelipatan
flight tidak sama dengan jumlah gate

%membentuk solusi awal
pop=zeros(ceil(f/g),g);
v=zeros(ceil(f/g),g);
if mod(f,g)==0
    for i=1:ceil(f/g)
    pop(i,:)=rand(1,g);%generate populasi awal
    v(i,:)=rand(1,g);%generate kecepatan awal
    end
```

```
else
    for i=1:floor(f/g)
        pop(i,:)=rand(1,g);%generate populasi awal
        v(i,:)=rand(1,g);%generate kecepatan awal
    end
    pop(ceil(f/g),1:(mod(f,g)))=rand(1,(mod(f,g)));
    v(ceil(f/g),1:(mod(f,g)))=rand(1,(mod(f,g)));
end

%mendapatkan flight yang hendak berada dalam 1 gate
q=zeros(ceil(f/g),g);
for i=1:ceil(f/g)
    [~,I]=sort(pop(i,:),'descend');
    q(i,:)=((i-1)*g)+I;
end
for i=(mod(f,g)+1):g
    if q(ceil(f/g),i)>f
        q(ceil(f/g),i)=f+1;
    end
end

%sesuaikan flight yang ada dalam 1 gate supaya hampir semua
feasible
k=1;
while k<ceil(f/g)
    l=1;
    while l<g+1
        B=x(q(k,l),q(k+1,l));
        if B==1
            l=l+1;
        else
            if l<g
                m=l+1;
                q(k+1,[l m])=q(k+1,[m l]);
                C=x(q(k,l),q(k+1,l));
                while m<g+1
                    q(k+1,[l m])=q(k+1,[m l]);
                    C=x(q(k,l),q(k+1,l));
                    if C<1 && m==g
                        [~,I]=max(q(ceil(f/g),:));
                        q(k+1,[l I])=q(k+1,[I l]);
                        q([k+1 ceil(f/g)],I)=q([ceil(f/g) k+1],I);
                        q(k+1,[l I])=q(k+1,[I l]);
                        m=m+1;
                    elseif C<1 && m<g
                        m=m+1;
                    else m=g+1;
                    end
                end
            else
                [~,I]=max(q(ceil(f/g),:));
                q(k+1,[l I])=q(k+1,[I l]);
                q([k+1 ceil(f/g)],I)=q([ceil(f/g) k+1],I);
                q(k+1,[l I])=q(k+1,[I l]);
            end
            l=l+1;
        end
    end
```

```
    if l<g
        k=k;
    else k=k+1;
    end
end
t=ceil(f/g);
u=1;
pena=ones(1,f);
pend=ones(1,f);
while u<g+1
    D=x(q(t,u),q(:,u));
    [W,Z]=min(D);
    if W<1
        if u<g
         w=u+1;
        while w<g+1
        q(t,[u w])=q(t,[w u]);
        D=x(q(t,u),q(:,u));
        [Y,Z]=min(D);
        if Y<1 && w==g
            if at(q(Z,u))<at(q(t,u)) && dt(q(Z,u))>dt(q(t,u))
                pena(q(t,u))=10;
                pend(q(t,u))=10;
            elseif at(q(Z,u))<at(q(t,u)) && dt(q(Z,u))<dt(q(t,u))
                pena(q(t,u))=10;
            elseif at(q(Z,u))>at(q(t,u)) && dt(q(Z,u))<dt(q(t,u))
                pena(q(Z,u))=10;
                pend(q(Z,u))=10;
            else pena(q(Z,u))=10;
            end
            w=w+1;
        elseif Y<1 && w<g
            w=w+1;
        else w=g+1;
        end
        end
        u=u+1;
        else
            if at(q(Z,u))<at(q(t,u)) && dt(q(Z,u))>dt(q(t,u))
                pena(q(t,u))=10;
                pend(q(t,u))=10;
            elseif at(q(Z,u))<at(q(t,u)) && dt(q(Z,u))<dt(q(t,u))
                pena(q(t,u))=10;
            elseif at(q(Z,u))>at(q(t,u)) && dt(q(Z,u))<dt(q(t,u))
                pena(q(Z,u))=10;
                pend(q(Z,u))=10;
            else pena(q(Z,u))=10;
            end
            u=u+1;
        end
    else u=u+1;
    end
end
%flight yang tidak feasible dikenakan penalti yang nantinya akan
%dikalkukasi untuk penghitungan tpwd

%menghapus fake flight sehingga solusi sesuai dengan jumlah flight
yang ada
for d=1:g
```

```
    for e=1:ceil(f/g)
        for r=1:ceil(f/g)
            for s=1:g
                if d==s && e==r
                    q(e,d)=q(e,d);
                elseif q(e,d)==f+1
                    q(e,d)=0;
                end
            end
        end
    end
end

%menghitung total walking distance dari masing-masing gate menuju
arriving
%hall dan departing hall
for m=1:g
    totalwdh(m)=wdga(m)+wdgd(m);
end
[~,GP]=sort(totalwdh);
%menghitung total passenger masing-masing gate
P=zeros(ceil(f/g),g);
TPG=zeros(1,g);
i=1;
while i<g+1
    for j=1:ceil(f/g)
        if q(j,i)==0
            P(j,i)=0;
        else P(j,i)=ap(q(j,i)) + dp(q(j,i));
        end
    end
    TPG(i)=sum(P(:,i));
    i=i+1;
end
[~,TP]=sort(TPG,'descend');%urutkan total passenger walking
distance dari yang terbesar
%membuat update peletakan gate supaya lebih optimal
q(:,[TP GP])=q(:,[GP TP]);

%menghitung fungsi tujuan
%transfer passenger walking distance
tpdist=zeros(f,f);
for i=1:f
    for j=1:f
        if tp(i,j)>0
            [~,K]=ind2sub(size(q),find(q==i));
            [~,L]=ind2sub(size(q),find(q==j));
            tpdist(i,j)=wdg(K,L);
        end
    end
end
transpwd=tpdist.*tp;
%arriving passenger walking distance
for i=1:f
    [~,J]=ind2sub(size(q),find(q==i));
    arrivaldist(i)=wdga(J,:)*pena(i);
end
arrivalpwd=arrivaldist.*ap;
```

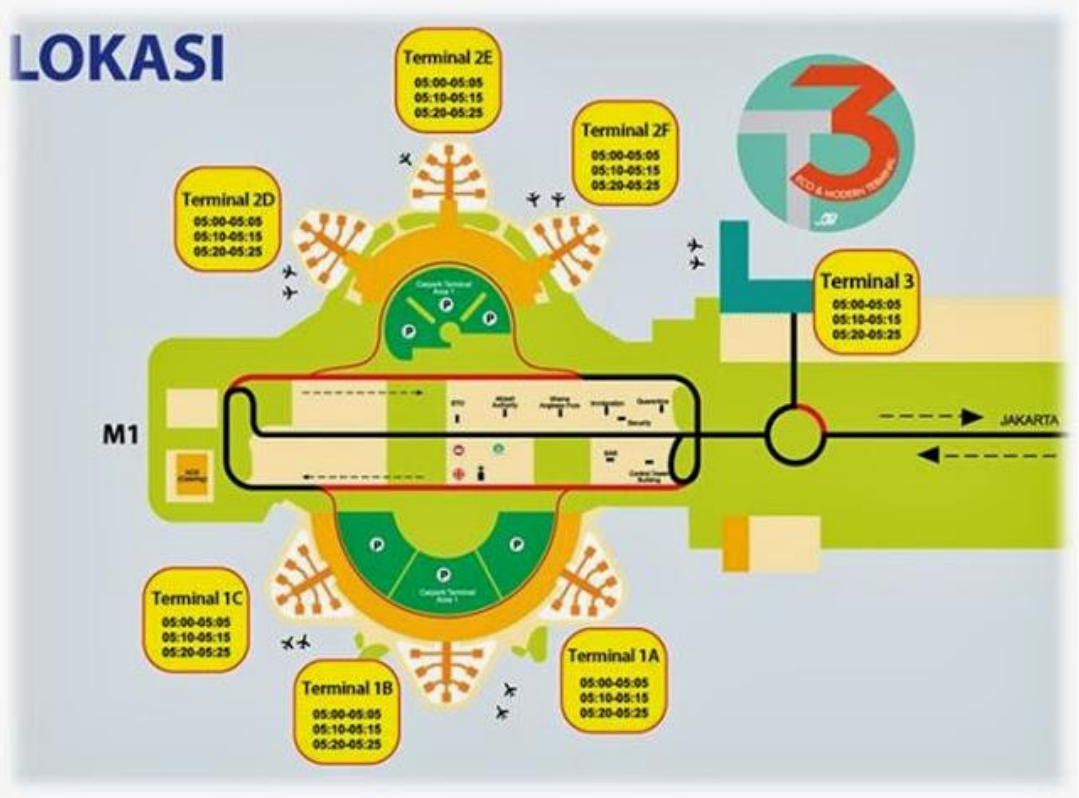
```
%departing passenger walking distance
for i=1:f
    [~,J]=ind2sub(size(q),find(q==i));
    departdist(i)=wdgd(J,:)*pend(i);
end
departpwd=departdist.*dp;
tpwd=(sum(sum(transpwd),2) + sum(arrivalpwd) +
sum(departpwd));%nilai tpwd
Pbest=pop;
Gbest=pop;
qq=q;
tpwds=tpwd;

%update populasi
rhomax=0.9;rhomin=0.1;it=1;
c1=0.1;c2=0.4;
popnew=pop;
while it<iterasi+1
    rho(it)=rhomax-((rhomax-rhomin)/iterasi)*it;
    for j=1:ceil(f/g)
    v(j,:)=(rho(it)*v(j,:))+(c1*rand).*(Pbest(j,:)-
pop(j,:))+(c2*rand).*(Gbest(j,:)-pop(j,:));
    popnew(j,:)=popnew(j,:).*v(j,:);
    end
    qnew=zeros(ceil(f/g),g);
    for i=1:ceil(f/g)
        [~,I]=sort(popnew(i,:),'descend');
        qnew(i,:)=((i-1)*g)+I;
    end
    for i=(mod(f,g)+1):g
    if qnew(ceil(f/g),i)>f
        qnew(ceil(f/g),i)=f+1;
    end
end

k=1;
while k<ceil(f/g)
    l=1;
    while l<g+1
        B=x(qnew(k,l),qnew(k+1,l));
        if B==1
            l=l+1;
        else
            if l<g
                m=l+1;
                while m<g+1
                    qnew(k+1,[l m])=qnew(k+1,[m l]);
                    C=x(qnew(k,l),qnew(k+1,l));
                    if C<1 && m==g
                        [~,I]=max(qnew(ceil(f/g),:));
                        qnew(k+1,[l I])=qnew(k+1,[I l]);
                        qnew([k+1 ceil(f/g)],I)=qnew([ceil(f/g)
k+1],I);
                        qnew(k+1,[l I])=qnew(k+1,[I l]);
                        m=m+1;
                    elseif C<1 && m<g
                        m=m+1;
                    else m=g+1;
```

```
                    end
                end
            else
                [~,I]=max(qnew(ceil(f/g),:));
                qnew(k+1,[l I])=qnew(k+1,[I l]);
                qnew([k+1 ceil(f/g)],I)=qnew([ceil(f/g) k+1],I);
                qnew(k+1,[l I])=qnew(k+1,[I l]);
            end
            l=l+1;
        end
    end
    if l<g
    else k=k+1;
    end
end
t=ceil(f/g);
u=1;
pena=ones(1,f);
pend=ones(1,f);
while u<g+1
    C=x(qnew(t,u),qnew(:,u));
    [W,Z]=min(C);
    if W<1
        if u<g
         w=u+1;
        while w<g+1
        qnew(t,[u w])=qnew(t,[w u]);
        D=x(qnew(t,u),qnew(:,u));
        [Y,Z]=min(D);
        if Y<1 && w==g
            if at(qnew(Z,u))<at(qnew(t,u)) &&
dt(qnew(Z,u))>dt(qnew(t,u))
                pena(qnew(t,u))=10;
                pend(qnew(t,u))=10;
            elseif at(qnew(Z,u))<at(qnew(t,u)) &&
dt(qnew(Z,u))<dt(qnew(t,u))
                pena(qnew(t,u))=10;
            elseif at(qnew(Z,u))>at(qnew(t,u)) &&
dt(qnew(Z,u))<dt(qnew(t,u))
                pena(qnew(Z,u))=10;
                pend(qnew(Z,u))=10;
            else pena(qnew(Z,u))=10;
            end
            w=w+1;
        elseif Y<1 && w<g
            w=w+1;
        else w=g+1;
        end
        end
        u=u+1;
        else
            if at(qnew(Z,u))<at(qnew(t,u)) &&
dt(qnew(Z,u))>dt(qnew(t,u))
                pena(qnew(t,u))=10;
                pend(qnew(t,u))=10;
            elseif at(qnew(Z,u))<at(qnew(t,u)) &&
dt(qnew(Z,u))<dt(qnew(t,u))
                pena(qnew(t,u))=10;
```

```
            elseif at(qnew(Z,u))>at(qnew(t,u)) &&
dt(qnew(Z,u))<dt(qnew(t,u))
                pena(qnew(Z,u))=10;
                pend(qnew(Z,u))=10;
            else pena(qnew(Z,u))=10;
            end
            u=u+1;
        end
    else u=u+1;
    end
end

for d=1:g
    for e=1:ceil(f/g)
        for r=1:ceil(f/g)
            for s=1:g
                if d==s && e==r
                    qnew(e,d)=qnew(e,d);
                elseif qnew(e,d)==f+1
                    qnew(e,d)=0;
                end
            end
        end
    end
end

P=zeros(ceil(f/g),g);
TPG=zeros(1,g);
i=1;
while i<g+1
    for j=1:ceil(f/g)
        if qnew(j,i)==0
            P(j,i)=0;
        else P(j,i)=ap(qnew(j,i)) + dp(qnew(j,i));
        end
    end
    TPG(i)=sum(P(:,i));
    i=i+1;
end
[~,TP]=sort(TPG,'descend');
qnew(:,[TP GP])=qnew(:,[GP TP]);

%menghitung fungsi tujuan update
%transfer passenger walking distance
tpdistnew=zeros(f,f);
for i=1:f
    for j=1:f
        if tp(i,j)>0
            [~,K]=ind2sub(size(qnew),find(qnew==i));
            [~,L]=ind2sub(size(qnew),find(qnew==j));
            tpdistnew(i,j)=wdg(K,L);
        end
    end
end
transpwdnew=tpdistnew.*tp;
%arriving passenger walking distance
for i=1:f
    [~,J]=ind2sub(size(qnew),find(qnew==i));
```

```
    arrivaldistnew(i)=wdga(J,:)*pena(i);
end
arrivalpwdnew=arrivaldistnew.*ap;
%departing passenger walking distance
for i=1:f
    [~,J]=ind2sub(size(qnew),find(qnew==i));
    departdistnew(i)=wdgd(J,:)*pend(i);
end
departpwdnew=departdistnew.*dp;
tpwdnew=sum(sum(transpwdnew),2) + sum(sum(arrivalpwdnew),2) +
sum(sum(departpwdnew),2);
if tpwdnew<tpwds
    tpwds=tpwdnew;
    pop=popnew;
    Pbest=popnew;
    Gbest=popnew;
    qq=qnew;
else
    Pbest=pop;
    pop=popnew;
end
it=it+1;
end

for i=1:g
    [Z,~]=sort(qq);
end
GAsolve=Z;
endtime=cputime;
time=endtime-starttime;
```

LAMPIRAN 2



Gambar Terminal di Bandara Soekarno – Hatta

Sumber : http://blog.qtiket.com/2012/01/daftar-terminal-bandara-soekarno-hatta.html

LAMPIRAN 3

Data Uji yang digunakan dalam penelitian

Flight	*Arrival*		*Departure*	
	***Time* (menit)**	***Passenger* (orang)**	***Time* (menit)**	***Passenger* (orang)**
1	0	341	55	285
2	30	159	290	153
3	30	164	290	151
4	300	329	395	345
5	345	178	415	156
6	375	281	465	329
7	385	154	430	180
8	450	322	505	283
9	450	317	505	309
10	495	214	610	207
11	505	315	560	319
12	505	286	560	295
13	515	144	555	149
14	525	204	675	198
15	540	179	595	200
16	585	163	705	146
17	610	216	670	179
18	610	340	705	294
19	615	281	690	314
20	615	320	690	295
21	630	214	675	213
22	640	167	685	172
23	675	218	735	207
24	680	201	810	209
25	715	318	785	300
26	750	213	795	189
27	750	204	825	217
28	755	135	820	126
29	755	134	820	143
30	800	195	825	192
31	800	290	855	292
32	800	340	855	294
33	800	178	865	185
34	800	189	860	181
35	810	152	870	178
36	815	192	870	194
37	815	289	935	330
38	815	216	870	183

Flight	*Arrival*		*Departure*	
	***Time* (menit)**	***Passenger* (orang)**	***Time* (menit)**	***Passenger* (orang)**
39	815	319	880	300
40	815	336	880	327
41	830	196	880	204
42	835	314	1080	297
43	835	281	1080	290
44	845	217	910	189
45	875	208	1090	179
46	895	211	945	198
47	895	201	945	185
48	915	183	1115	218
49	940	310	990	287
50	940	285	990	348
51	945	328	1065	328
52	965	331	1020	310
53	965	307	1020	334
54	975	199	1020	189
55	985	159	1035	170
56	1015	316	1080	289
57	1015	300	1080	312
58	1015	296	1165	347
59	1050	202	1105	214
60	1050	218	1190	187
61	1080	325	1240	291
62	1080	207	1185	191
63	1085	217	1145	214
64	1085	301	1145	316
65	1090	304	1165	289
66	1145	179	1190	219
67	1155	207	1235	208
68	1155	220	1235	219
69	1160	157	1205	160
70	1165	327	1220	311
71	1165	288	1220	282
72	1180	219	1240	178
73	1235	350	1325	314
74	1235	332	1325	305
75	1255	220	1365	201
76	1270	186	1350	205
77	1280	217	1320	184
78	1335	221	1430	186
79	1340	283	1410	283

Data Penumpang Transfer (orang)

Flight	4	8	9	10	11	12	13	15	41	42	43	44	62	63	64	65	67	68	69	70	71	72	73	74	75	76
2	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	15	25	10	8	13	28	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	20	18	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0
59	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0
61	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0
62	0	0	0	0	0	0	0	0	0	0	0	0	0	11	23	33	0	0	0	0	0	0	0	0	0	0
63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	60	0	0	0	0	0	0	0	0
64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	24	0	0	0	0	0	0	0	0
65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	22	0	0	0	0	0	0	0	0
66	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	15	0	0	0	0	0	0	0	0
67	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	10	15	10	4
68	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	13	0	0	0	16	13

Matriks Jarak antar *Gate* (skala 1:100 meter)

Gate	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	2.5	3.5	3.5	3.5	3	3	11	11.5	12	12	12	11.5	11
2	2.5	0	2.5	2.5	2.5	2	3	11.5	12	12.5	12.5	12.5	12	11.5
3	3.5	2.5	0	1	1	2.5	3.5	12	12.5	13	13	13	12.5	12
4	3.5	2.5	1	0	1	2.5	3	12	12.5	13	13	13	12.5	12
5	3.5	2.5	1	1	0	2.5	3	12	12.5	13	13	13	12.5	12.5
6	3	2	2.5	2.5	2.5	0	2.5	11.5	12	12.5	12.5	12.5	12	11.5
7	3	3	3.5	3	3	2.5	0	11	11.5	12	12	12	11.5	11
8	11	11.5	12	12	12	11.5	11	0	2.5	3.5	3.5	3.5	3	3
9	11.5	12	12.5	12.5	12.5	12	11.5	2.5	0	2.5	2.5	2.5	2	3
10	12	12.5	13	13	13	12.5	12	3.5	2.5	0	1	1	2.5	3.5
11	12	12.5	13	13	13	12.5	12	3.5	2.5	1	0	1	2.5	3
12	12	12.5	13	13	13	12.5	12	3.5	2.5	1	1	0	2.5	3
13	11.5	12	12.5	12.5	12.5	12	11.5	3	2	2.5	2.5	2.5	0	2.5
14	11	11.5	12	12	12	11.5	11	3	3	3.5	3	3	2.5	0

Data Jarak *Gate* dengan *Arrival* dan *Departure Hall* (skala 1:100 meter)

Gate	*Arrival Hall*	*Departure Hall*
1	5.5	3.5
2	6	4
3	6.5	4.5
4	6.5	4.5
5	6.5	4.5
6	6	4
7	5.5	3.5
8	7.5	3.5
9	8	4
10	8.5	4.5
11	8.5	4.5
12	8.5	4.5
13	8	4
14	7.5	3.5

BIOGRAFI PENULIS

Hendrik Febriyanto, lahir di Bojonegoro 16 Februari 1994. Menempuh pendidikan di TK Mardisiwi, SD Negeri 1 Sumberrejo, SMP Negeri 1 Sumberrejo, SMA Negeri 1 Bojonegoro dan Teknik Industri Institut Teknologi Sepuluh Nopember (ITS) Surabaya.

Selama masa studi pernah aktif berorganisasi seperti OSIS semasa SMP dan SMA. Sedangkan kala menempuh studi S-1 juga aktif dalam organisasi Unit Kegiatan Mahasiswa (UKM). Pernah menjabat sebagai Ketua UKM Bridge ITS Periode 2014/2015 dimana sebelumnya menjadi Kepala Bidang Perlengkapan Periode 2013/2014. Selain di UKM juga terlibat aktif dalam pengembangan UKM di ITS melalui Lembaga Minat Bakat (LMB) dengan menjadi staff Bidang Event Periode 2013/2014 dan menjadi Kepala Departemen Domestik Periode 2015/2016.

Selain aktif dalam kampus, penulis juga terlibat aktif dalam organisasi bridge daerah yakni sebagai Staff Bidang Pembinaan dan Prestasi Gabungan Bridge Kabupaten Bojonegoro Periode 2013-2016, dan sebagai Staff Bidang Hubungan Masyarakat Gabungan Bridge Jawa Timur Periode 2017-2020.

Saat ini penulis aktif dalam kegiatan olahraga bridge baik regional maupun nasional. Selain itu juga aktif sebagai supervisor sebuah lembaga bimbingan belajar Rumah Belajar RHEMA dari tahun 2016 sampai sekarang. Saat ini penulis sedang mengembangkan sebuah program *scoring* untuk turnamen bridge supaya mampu dilihat secara daring dan *easy going* untuk dipergunakan.

Penulis dapat dapat dihubungi melalui

email : hendrik_febriyanto@yahoo.com

whatsapp : +62 89677523113